UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Everton Luiz Vieira

USO DO CONCEITO DE PASSOS OBRIGATÓRIOS PARA APRIMORAR O PROCESSO DE CONTAGEM DO MÉTODO "PONTOS DE CASO DE USO"

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Raul Sidnei Wazlawick
Orientador

Florianópolis, Junho de 2007.

USO DO CONCEITO DE PASSOS OBRIGATÓRIOS PARA APRIMORAR O MÉTODO "PONTOS DE CASO DE USO"

Everton Luiz Vieira

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação na área de concentração de Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Rogério Cid Bastos

Banca Examinadora:

_____

Raul Sidnei Wazlawick (orientador)

_____

Itana Maria de Souza Gimenes

_____

Frank Siqueira

_____

Ricardo Pereira e Silva

"Vencer-se a si próprio é a maior das vitórias."

(Platão)

Dedico aos meus pais Maria Merces e Verner, minha avó Benta, meus irmãos Elaine, Eduardo e Leomar e a minha esposa Juliana, que colaboraram com a realização e apoio para a conclusão do mestrado.

AGRADECIMENTOS

Queria agradecer a todos que ao longo do mestrado me ajudaram e permaneceram ao meu lado em diversos momentos.

Aos meus pais Maria Merces e Verner, a minha avó Benta e aos meus irmãos Elaine, Eduardo e Leomar que despenderam desde suas emoções ao me verem entrando no mestrado até o apoio financeiro quando precisei.

A minha esposa Juliana, seus pais Maria Rosa e Roberto e a minha cunhada Roberta que receberam em sua casa com carinho e tiveram compreensão nos momentos de dificuldade.

Aos amigos: Marcos, Zene e seus filhos Rodrigo e Vinícius pelas festas em sua casa, e aos amigos Luiz, Tânia e Rafael (Kerubim), aos meus padrinhos de casamento Kuesley e Priscila que me ajudaram a esquecer um pouco o mestrado por algumas horas de distração.

Ao professor e amigo Harry Erwin Moissa que me incentivou a fazer o mestrado e me apoiou durante todo o andamento.

Ao professor e orientador Raul Sidnei WAZLAWICK que me conduziu durante o mestrado e mostrou ser uma ótima pessoa tanto quanto o profissional respeitado por todos.

Ao pessoal da secretaria do PPGCC, principalmente à Vera Lúcia Sodré Teixeira (Verinha) que sempre me recepcionou e ajudou no que foi preciso.

E a CAPES, fundação do Ministério da Educação, que ajudou financeiramente boa parte do mestrado com uma bolsa de pesquisa.

SUMÁRIO

Listas de Figuras ..... **ix**

Lista de Quadros ..... **x**

Resumo ..... **xi**

Abstract ..... **xii**

1 Introdução **1**

1.1 Apresentação do Trabalho 1

1.2 Justificativa 2

1.3 Objetivos do Trabalho 3

1.3.1 Objetivo Geral 3

1.3.2 Objetivos Específicos 3

1.4 Metodologia 4

1.5 Hipótese 4

1.6 Resultados Esperados 5

1.7 Estrutura da Dissertação 5

2 Revisão da Literatura **7**

2.1 Casos de Uso 7

2.1.1 O que é um Caso de Uso 8

2.1.2 Escrevendo Casos de Uso 8

2.1.2.1 Casos de Uso CRUD 9

2.1.3 Expansão dos Casos de Uso 10

2.1.4 Descrição Essencial e Real do Caso de Uso 15
2.1.5 Escrevendo Casos de Uso para o propósito de estimativa..... 15

2.2 Métricas de Estimativas de Software..... 16
2.2.1 Linhas de Código 16
2.2.2 Análise de Pontos de Função 17
2.2.2.1 Trabalhos relacionados à Análise de Pontos de Função 19
2.2.3 Análise de Pontos de Função Mk II 20
2.2.4 Pontos de Caso de Uso 22
2.2.4.1 Contagem dos Pontos de Caso de Uso..... 23
2.2.4.2 Produtividade com o método Pontos de Caso de Uso 26
2.2.4.3 Contagem dos Casos de Uso Estendidos e Incluídos 27
2.2.4.4 Problemas com a contagem dos casos de uso e o método Pontos de Caso de Uso 28
2.2.4.5 Trabalhos relacionados a Pontos de Caso de Uso 28
2.2.4.6 Outros trabalhos..... 30

3 Teoria para Estimativa de Software Baseada em Casos de Uso no Desenvolvimento Orientado a Objetos 33

3.1 Primeira Etapa: Listagem dos Casos de Uso na Fase de Concepção 34
3.1.1 Estimativa usando apenas o número de casos de uso..... 35
3.1.2 Acrescentando uma medida de complexidade aos casos de uso..... 36
3.1.3 Separando fatores dependentes e independentes dos casos de uso. 37
3.1.4 Considerando que a análise inicial não é completa. 37

3.2 Segunda Etapa: Expansão dos Casos de uso..... 39

3.3 Terceira Etapa: Contratos 41

3.4 A Fase de Projeto e Implementação 43

4 Avaliação 45

4.1 Estudo de Caso 1 – Avaliação da Contagem de Passos 45

4.2 Estudo de Caso 2 – Avaliação da Subestimação 51

4.3 Estudo de Caso 3 - Avaliação do Tempo de Desenvolvimento por Caso de Uso 53

5 Conclusões 56

5.1 Limitações do Trabalho 57

5.2 Trabalhos Futuros 58

Referências Bibliográficas 59

Referências Consultadas 61

Apêndice 63

Anexo 1 72

Anexo 2 74

Listas de Figuras

Figura 1 - Caso de Uso: Sistema Videolocadora. 11

Figura 2 - Caso de Uso: Sistema Videolocadora. 14

Figura 3 - Contagem de Todos os Passos..... 49

Figura 4 - Contagem Somente dos Passos Obrigatórios 49

Lista de Quadros

Quadro 1 - Fatores de Ajuste Técnico 22

Quadro 2 - Classificação de Ator 24

Quadro 3 - Classificação do Caso de Uso 24

Quadro 4 - Fatores de Complexidade Técnica 25

Quadro 5 - Fatores Ambientais 25

Quadro 6 - Caso de Uso Emprestar Livros 46

Quadro 7 - Caso de Uso Emprestar Livros (Passos Obrigatórios) 47

Quadro 8 - Sistema Biblioteca 48

Quadro 9 - Sistema Hotel 48

Quadro 10 - Análise dos Sistemas 52

Resumo

Métricas de software são essenciais para estimar o tempo e o custo do desenvolvimento de software. Diversos métodos propõem formas de contagem para gerar as estimativas do tempo real de desenvolvimento, como, por exemplo, o método "Pontos de Caso de Uso". O principal problema deste método está na falta de precisão na estimativa de complexidade de um caso de uso. O método da contagem de passos (PCU) pode gerar muita discrepância entre diferentes analistas, os quais possivelmente irão gerar estimativas diferentes. Este trabalho apresenta uma técnica para classificação dos passos nos casos de uso baseada em uma hipótese razoável, confirmada por estudos de caso, que gerou redução na discrepância da estimativa em relação ao método PCU original. A razoabilidade desta hipótese é sugerida por um conjunto de definições (teoria) que determina a estimativa de esforço real ao longo das várias fases do processo de análise do desenvolvimento de software.

Abstract

Software metrics are essential in order to discover the cost and time of software development. There exist a number of counting approaches to generate estimates for the real development time, as for example “Use Case Points”. The main problem with this method particularly is the lack of precision in the estimation of a use case complexity. The “counting of steps” method (UCP) can generate discrepancy among different analysts, which will possibly provide significantly different estimations. This work presents a new technique for classification of steps in use cases based on a reasonable hypothesis, confirmed by case studies. This technique allowed a reduction in discrepancy on estimations made by different analysts in comparison to the original UCP method. The reasonable hypothesis is presented as a set of definitions (theory) that intends to capture the explanation of the real effort estimation throughout the activities of analysis of the software development.

1 Introdução

Estimativas são necessárias para determinar quando é viável desenvolver o sistema e propor contratos. Além disso, boas estimativas de esforço são necessárias para planejar e gerenciar projetos de software (PRESSMAN, 2001). Diversos modelos para estimativa de software foram propostos a partir da década de 60, como Linhas de Código, Análise de Pontos por Função e mais recentemente, Pontos de Caso de Uso, proposto por KARNER (1993) para estimar o desenvolvimento de software orientado a objetos a partir dos casos de uso levantados na fase de requisitos.

KARNER (1993) propôs o método Pontos de Caso de Uso (PCU) como uma melhoria para a Análise de Pontos por Função (APF), que inicialmente foi proposta para estimar o desenvolvimento de softwares estruturados. Devido o fato do método PCU ser uma evolução do método APF, alguns problemas foram adquiridos e incorporados. Diversas melhorias foram propostas para tornar o método mais preciso. O trabalho mais recente e importante foi o de RIBU (2002), em uma dissertação de mestrado da Universidade de Oslo.

Mesmo com as melhorias propostas, o método PCU ainda possui problemas inerentes aos casos de uso, que são a base para gerar as estimativas de software. Os casos de uso são definidos com passos descritos subjetivamente e não existe uma padronização predominantemente adotada para descrevê-los. Portanto, diferentes analistas poderão descrever os mesmos casos de uso com relativamente poucos ou muitos passos.

1.1 Apresentação do Trabalho

Este trabalho apresenta uma melhoria para a descrição dos casos de uso, com base na proposta de WAZLAWICK (2004), que classifica os passos dos casos de uso e determina quais passos efetivamente agregam valor no esforço para o desenvolvimento de software orientado a objetos, demonstrando que diferentes casos de uso podem conter os mesmos passos descritos por diferentes analistas, tornando-os assim, menos discrepantes e gerando estimativas mais precisas.

Além disso, é proposta uma nova teoria para gerar estimativas do tempo de desenvolvimento de software orientado a objetos com base nos casos de uso, utilizando o processo de desenvolvimento de software de LARMAN (2002) adaptado por WAZLAWICK (2004).

1.2 Justificativa

Cada vez mais as empresas de desenvolvimento de software necessitam gerar estimativas para definir contratos com seus clientes. Existem diversos métodos que ajudam a gerar estas estimativas. O mais utilizado até o momento é o método Análise de Pontos de Função – APF, que já foi modificado e refinado diversas vezes, desde sua concepção nos anos 70. Trata-se de um método consolidado, que foi inicialmente definido para aplicações com processamento de dados e linguagens estruturadas.

Porém, a APF não tem alcançado os mesmos resultados para gerar estimativas para o desenvolvimento de software orientado a objetos (SPARKS & KASPCZYNSKI, 1999). Além disso, os conceitos da orientação a objetos conforme RULE (2001) não podem ser convertidos em conceitos abordados pela APF (por exemplo, elementos de entradas e arquivos lógicos).

Em 1993, surge um novo método chamado “Pontos de Caso de Uso”, criado por Gustav Karner, como uma adaptação da APF para gerar estimativas para o desenvolvimento de software orientado a objetos. O método Pontos de Caso de Uso é dependente de casos de uso, e estes são definidos por descrições livres e que podem ser ambíguas. Não existe padrão formal predominante para descrever os casos de uso e muitas vezes em uma mesma empresa são escritos de forma diferente, ou seja, analistas diferentes produzem estimativas diferentes para o mesmo projeto.

Além disso, o método proposto inicialmente por KARNER (1993) e mais recentemente aperfeiçoado por RIBU (2002), ainda possui limitações:

a) O método PCU foi criado há mais de 10 anos, mas não é totalmente confiável e ainda produz estimativas discrepantes.

b) Não há uma forma definitiva de propor a quantidade de horas/equipe por caso de uso.

c) Existem poucos trabalhos sobre Pontos de Caso de Uso. Essa pode ser uma das razões de porque o método ainda não se tornou popular.

Na tentativa de melhorar o método e atingir melhores resultados para as estimativas, este trabalho aplica uma classificação dos passos nos casos de uso, que conduzirá os diferentes analistas a descrever a mesma quantidade ou uma variação mais baixa dos passos que geram o efetivo esforço de desenvolvimento em software orientado a objetos.

1.3 Objetivos do Trabalho

1.3.1 Objetivo Geral

O principal objetivo do trabalho é aperfeiçoar o método Pontos de Caso de Uso, para gerar estimativas mais precisas por diferentes analistas. Fazendo desta forma com que diferentes analistas gerem estimativas mais próximas do tempo real de desenvolvimento de software orientado a objetos.

1.3.2 Objetivos Específicos

Este trabalho tem como objetivos específicos:

a) Testar um método, a fim de padronizar as descrições dos casos de uso, permitindo produzir menos discrepância nos resultados das estimativas com o método Pontos de Caso de Uso.

b) Aperfeiçoar o método Pontos de Caso de Uso a fim de gerar estimativas mais precisas do tempo real do desenvolvimento de software orientado a objetos.

1.4 Metodologia

Para a realização deste trabalho foi feito um estudo bibliográfico para obter o entendimento teórico necessário para a definição dos conceitos sobre métricas de software, os métodos de estimativas e seus respectivos processos de contagem, bem como os trabalhos correlatos, focando principalmente o método Pontos de Caso de Uso.

Neste estudo inicial, foi identificado o modelo de descrição dos casos de uso utilizado por WAZLAWICK (2004) e adaptado para o processo de contagem do método Pontos de Caso de Uso original. Este modelo de descrição consiste na classificação dos passos dos casos de uso em eventos e respostas de sistema e tende a gerar estimativas menos discrepantes por diferentes analistas, já que os mesmos tendem a descrever os mesmos passos obrigatórios nos casos de uso, ou seja, os passos que efetivamente geram esforço. Este estudo foi aplicado em dois estudos de caso e os resultados são apresentados no capítulo 5.

A partir da aplicação do processo de desenvolvimento de LARMAN (2002) adaptado por WAZLAWICK (2004) no desenvolvimento de um software orientado a objetos, foi definida uma teoria que apresenta uma fórmula para cada etapa do processo de desenvolvimento, gerando uma estimativa conforme os artefatos gerados na fase de análise. Esta teoria foi aplicada em um estudo de caso para gerar os valores das variáveis que vão produzir as estimativas de tempo.

1.5 Hipótese

Este trabalho tem como hipótese o uso de uma descrição padronizada de caso de uso, como a proposta por WAZLAWICK (2004), que diferencia passos de contexto de entrada e saída de informação (evento e resposta de sistema). Desta forma, espera-se gerar estimativas menos discrepantes do que os atuais métodos que contabilizam o número de passos sem diferenciá-los.

1.6 Resultados Esperados

Com este estudo espera-se a padronização na descrição dos passos obrigatórios dos casos de uso, tornando a quantidade das transações ou passos menos discrepantes, quando descritos por diferentes analistas.

Espera-se também que, com a aplicação da metodologia, seja reduzido o desvio padrão entre as estimativas e que sejam geradas estimativas cada vez mais precisas, mesmo sendo geradas por analistas pouco experientes.

Além disso, através da teoria proposta, espera-se que as estimativas possam ser geradas em qualquer etapa da fase de análise do processo de desenvolvimento de software orientado a objetos, à medida que os artefatos forem gerados.

1.7 Estrutura da Dissertação

Este trabalho consiste de seis capítulos.

No capítulo 2 é realizada a revisão da literatura, onde são apresentados os conceitos para casos de uso e as técnicas para descrevê-los, bem como a sugestão de WAZLAWICK (2004). Por fim, serão apresentados os principais conceitos sobre métricas de estimativas de software. Conseqüentemente, serão apresentados os principais métodos de estimativa, processo de contagem e trabalhos relacionados.

No capítulo 3 é apresentada a Teoria para Estimativa de Software Baseada em Casos de Uso no Desenvolvimento Orientado a Objetos, que consiste na teoria proposta neste trabalho para gerar estimativas mais precisas, a partir da utilização do processo de desenvolvimento de software de WAZLAWICK (2004).

No capítulo 4 são apresentados os materiais que foram utilizados para realizar as pesquisas e os métodos utilizados para gerar os resultados.

No capítulo 5 são mostrados os resultados a partir dos estudos do capítulo 4.

Finalmente, no capítulo 6 são apresentadas as conclusões deste trabalho, incluindo as conclusões sobre a utilização da sugestão de WAZLAWICK (2004) na descrição dos passos nos casos de uso e as conclusões sobre a teoria proposta neste trabalho. Além

disso, são sugeridos alguns trabalhos futuros, que possam complementar e/ou melhorar este estudo.

2 Revisão da Literatura

Nesta seção são apresentados os principais conceitos sobre casos de uso e técnicas para descrevê-los, logo após os conceitos sobre métricas de estimativas de software e conseqüentemente os principais métodos de estimativa, processo de contagem e trabalhos relacionados.

2.1 Casos de Uso

Ivar Jacobson inventou em 1960, quando trabalhava na Ericsson, o que mais tarde se tornou conhecido como casos de uso. No final dos anos 80, Jacobson apresentou seu trabalho à comunidade, onde os casos de uso tornaram-se parte significante no processo de engenharia de requisitos (COCKBURN, 2005). Jacobson deixou a Ericsson em 1987 e estabeleceu a *Objectory* AB em Estocolmo, quando ele e seus associados desenvolveram um processo de software chamado *Objectory* – uma abreviação de *Object Factory* ou Fábrica de Objetos, uma técnica de diagramação desenvolvida para o conceito de caso de uso (RIBU, 2001). Em 1992, Jacobson inventou a metodologia de software OOSE (*Object Oriented Software Engineering* ou Engenharia de Software Orientada a Objetos), uma metodologia dirigida por casos de uso, na qual os casos de uso são envolvidos em todos os estágios de desenvolvimento, o qual inclui: análise, projeto, validação e teste (RIBU, 2001).

Em 1994, Alistair Cockburn construiu o Modelo Conceitual de Atores e Objetivos enquanto escrevia regras de caso de uso para a *IBM Consulting Group*. Ele forneceu um guia de como estruturar e escrever casos de uso. Conforme COCKBURN (2005), "de 1994 a 1999, as idéias permaneceram estáveis, embora houvessem algumas conclusões soltas na teoria". COCKBURN (2005) ainda ressaltou que "a *UML – Unified Modeling Language* teve pouco impacto nas suas idéias e vice-versa, embora o grupo de padrões UML tenha uma forte influência de ferramentas de edição de diagramas, de forma que a natureza textual dos casos de uso foi perdida no padrão".

2.1.1 O que é um Caso de Uso

Caso de uso é uma técnica amplamente utilizada na indústria para descrever e capturar os requisitos funcionais de software (RIBU, 2001).

Segundo COCKBURN (2005), os casos de uso são fundamentalmente uma forma textual, embora possam consistir em fluxogramas ou diagramas de seqüência, redes de Petri ou linguagens de programação. Servem como meio de comunicação entre pessoas que muitas vezes não possuem treinamento especial e portanto, texto simples geralmente é a melhor escolha.

COCKBURN (2005) relata que:

> "Um caso de uso bem escrito é fácil de ler. O caso de uso contém sentenças em uma única forma gramatical – um passo de ação simples – na qual um ator alcança um resultado ou transmite informação para outro ator. Aprender a ler casos de uso não deve tomar mais do que poucos minutos. Aprender a escrever um bom caso de uso é difícil".

Segundo WAZLAWICK (2004, p. 44), os casos de uso devem corresponder aos principais processos de negócio da empresa e possivelmente são associados a vários requisitos funcionais.

2.1.2 Escrevendo Casos de Uso

> "Toda organização captura requisitos para satisfazer suas necessidades. Há até mesmo padrões disponíveis para descrições dos requisitos. Em qualquer deles, casos de uso ocupam somente uma parte do total de requisitos documentados. Eles não são todos os requisitos - são apenas os requisitos comportamentais, contudo são todos os comportamentais (COCKBURN, 2005).

Cada caso de uso contém um conjunto de requisitos funcionais do sistema. Na fase de concepção de um sistema, os casos de uso são listados e escritos em alto nível, listando os atores e os requisitos correlacionados (WAZLAWICK, 2004). A descrição de alto nível consiste em apenas um parágrafo que explica sucintamente o objetivo do caso de uso. Eventualmente algumas exceções dignas de nota podem ser também informadas, dando origem a um caso de uso com estilo "casual" (WAZLAWICK, 2004, p.64-65).

Conforme COCKBURN (2005), para escrever casos de uso como requisitos é preciso manter duas coisas em mente:

a) **Eles realmente são requisitos**: escrever exatamente os requisitos comportamentais que o sistema deve fazer;

b) **Eles não são todos os requisitos**: não especificar interfaces externas, formato de dados, regras de negócio e fórmulas complexas. Casos de uso ocupam somente uma fração muito importante dos requisitos que é necessário coletar.

Conforme WAZLAWICK (2004) deve-se tomar alguns cuidados para não definir processos muito longos ou muito triviais com casos de uso, seguindo as regras abaixo para descrever um caso de uso:

a) Deve ser monossessão, ou seja, executado em uma única interação e não se estendendo por vários dias.

b) Deve ser interativo, com informações fluindo para dentro e fora do sistema. Operações elementares de cadastro dificilmente serão casos de uso, pois são efetuadas normalmente em um único passo.

c) Deve produzir uma alteração consistente na informação armazenada. Consultas não devem ser consideradas casos de uso, pois apenas exibem os dados já existentes com alguma formatação específica.

2.1.2.1 Casos de Uso CRUD

Os casos de uso CRUD, representativas operações em inglês *Create*, *Retrieve*, *Update* e *Delete* ou Criar, Recuperar, Atualizar e Deletar (COCKBURN, 2005).

COCKBURN (2005) relata que eles fazem parte de um caso de uso muito maior chamado Gerenciar <Cacho>, que <Cacho> por sua vez, substitui qualquer elemento do domínio <Aluno>, <Cliente>, <Produto> que necessita das operações CRUD.

Segundo COCKBURN (2005), os autores se dividem com opiniões a respeito da melhor maneira de lidar com casos de uso CRUD. “Em princípio, eles são separados porque cada um é um objetivo separado, possivelmente realizado por uma pessoa

diferente com um nível de segurança diferente. No entanto, eles amontoam os casos de uso e podem triplicar o número de itens para rastrear". COCKBURN (2005), conta que começa com o caso de uso Gerenciar <Cacho> para ter a vantagem de uma menor desordem. Se a escrita torna-se complexa, ele quebra em diversas partes.

Segundo WAZLAWICK (2004), alterações como estas seguem uma mesma forma geral e consistem em um único acesso ao sistema. Não existe muita praticidade em considerar estas operações como casos de uso. Estes conceitos podem ser especificados de uma forma mais simples na fase de análise como meras operações de sistema, a não ser que exijam interatividade.

2.1.3 Expansão dos Casos de Uso

Após a descoberta dos casos de uso, já na fase de elaboração, são feitas as expansões dos casos de uso, o que corresponde ao aprofundamento da análise de requisitos e consiste no detalhamento do processo de negócio (WAZLAWICK, 2004).

Nesta etapa deve-se descrever passo a passo como o caso de uso ocorre, como é a interação entre os usuários e o sistema. A expansão dos casos de uso começa descrevendo passo a passo a interação entre os usuários e o sistema, criando o Fluxo Principal que é conhecido como o caminho feliz, no qual tudo dá certo, porque não são descritas as exceções. Após criar o Fluxo Principal são descritas as interações para os casos nos quais ocorre alguma exceção, os chamados Fluxos Alternativos, que são os procedimentos para contornar os problemas do Fluxo Principal (WAZLAWICK, 2004, p.60-69).

A Figura 1 apresenta um caso de uso expandido com o Fluxo Principal e os Fluxos Alternativos:

Fluxo Principal	Fluxos Alternativos
1. O cliente chega ao balcão com as fitas que deseja locar. 2. O cliente informa seu nome e entrega as fitas ao funcionário. 3. O funcionário registra o nome do cliente e inicia a locação. 4. O funcionário registra cada uma das fitas. 5. O funcionário finaliza a locação, devolve as fitas ao cliente e lhe informa a da data de devolução e o valor total da locação. 6. O cliente deixa a locadora com as fitas.	**3a. O cliente não possui cadastro** 3a.1 O cliente deve informar seus dados para cadastro. 3a.2 O funcionário registra o cadastro. 3a.3 Retorna ao Fluxo Principal no passo 3. **3b. O cliente possui pendências no cadastro (locação anterior não foi paga)** 3b.1 O cliente paga seu débito. 3b.2 O funcionário registra o pagamento, eliminando a pendência. 3b.3 Retorna ao Fluxo Principal no passo 3. **4a. A fita está reservada para outro cliente** 4a.1 O funcionário informa que a fita não está disponível para locação. 4a.2 Prossegue a locação no Fluxo Principal no passo 4 sem incluir a fita reservada. **4b. A fita está danificada** 4b.1 O funcionário informa que a fita está danificada. 4b.2 O funcionário registra que a fita está danificada. 4b.3 O funcionário verifica se existe outra fita disponível com o mesmo filme. 4b.3 Se existir, o funcionário substitui a fita e Retorna ao Fluxo Principal no passo 4, sem registrar a fita danificada.

FIGURA 1 - CASO DE USO: SISTEMA VIDEOLOCADORA (WAZLAWICK,2004).

Além do Fluxo Principal e Fluxo Alternativo, a seguir são apresentadas resumidamente algumas partes dos casos de uso (COCKBURN, 2005):

a) **Ator**: alguém ou algo com comportamento (COCKBURN, 2005). Uma lista das entidades que interagem com o sistema por meio do caso de uso, que podem ser tanto pessoas como sistemas externos (WAZLAWICK, 2004).

b) **Interessado**: alguém ou algo com interesse no comportamento do sistema (COCKBURN, 2005). Nem sempre só os atores são interessados no caso de uso. Um setor de contabilidade pode estar interessado nas vendas da empresa ou o

estoque pode querer saber quais produtos foram vendidos, ou seja, direta e indiretamente interessadas ao caso de uso (WAZLAWICK, 2004).

c) **Ator Primário**: o interessado que inicia uma interação com o sistema para alcançar um objetivo.

d) **Caso de Uso**: um contrato sobre o comportamento do sistema.

e) **Escopo**: identifica o sistema que está sendo discutido.

f) **Pré-Condições e Garantias**: o que deve ser verdadeiro antes e após a execução do caso de uso.

g) **Cenário de Sucesso Principal (Fluxo Principal)**: um caso de uso em que nada acontece de errado.

h) **Extensões (Fluxos Alternativos)**: o que pode ocorrer de diferente durante aquele cenário.

i) **Variações Tecnológicas e de Dados**: quase sempre isto ocorre nos dados que devem ser capturados, porque a captura dos dados pode acontecer de diversas formas como no caso do cliente identificar-se no banco com cartão magnético, leitura da íris e impressão digital. Estas variações não são Fluxos Alternativos.

Os analistas têm dificuldades em decidir o que pode e/ou deve ser colocado como passo do caso de uso expandido. De fato, duas pessoas que descrevem um mesmo processo, quase sempre definirão uma seqüência de passos diferentes (WAZLAWICK, 2004, p.67). Portanto, WAZLAWICK (2004) sugeriu uma forma de descrição dos casos de uso, nomeado como passos obrigatórios. Estes passos obrigatórios incluem os passos que indicam de alguma forma uma troca de informação entre os atores e o sistema. Ele afirmou ainda que todo o caso de uso possui passos obrigatórios, porque sem estas etapas, o caso de uso não pode prosseguir corretamente. Entretanto, os mesmos passos obrigatórios podem aparecer em diferentes posições de um caso de uso. Quando diferentes analistas descrevem um caso de uso, eles tendem a usar os mesmos passos obrigatórios, mas eles podem usar uma ordem diferente.

Existem dois tipos de passos obrigatórios:

a) Eventos de Sistema: passos que mostram alguma informação indo de um ator para o sistema.

b) Respostas do Sistema: passos que mostram alguma informação vindo do sistema para o(s) ator(es).

WAZLAWICK (2004) sugere que os Eventos de Sistema possam ser identificados no texto do caso de uso com o marcador [EV] e as Respostas do Sistema com o marcador [RS].

Os eventos de sistema devem conter informações (dados) que o sistema ainda não tem. Por exemplo, quando um usuário registra um produto em um sistema de venda, ele/ela está passando a informação que o sistema não tem: um produto que está sendo vendido. Esta informação tem que ser registrada em algum lugar na base de dados persistente pelo sistema (embora a tecnologia não seja relevante neste estágio da descrição do sistema - essencial). Por outro lado, quando um usuário pressiona um botão para ir da janela principal para uma janela de opção da venda (isto pode se escrito na forma essencial como "o usuário acessa as opções de vendas"), isto não é um evento relevante porque não existe troca de informação. Não existe a necessidade para o sistema registrar isto na base de dados. Isto é somente um passo navegacional e tem muito pouco impacto no esforço para desenvolver um sistema comparado ao esforço para desenvolver as transações baseadas nos eventos de sistema (WAZLAWICK, 2004).

As respostas de sistema também têm que trazer informação do sistema (possivelmente armazenados na base de dados) que o(s) ator(es) ainda não tem. Por exemplo, o retorno da informação que o cliente tem um limite para gastar é uma "resposta de sistema", porque esta informação está armazenada no sistema e os atores não necessariamente sabem como isto é atualizado. Por outro lado, os retornos de sistema do tipo "OK" (que representa somente um retorno possível para um evento de sistema) não são respostas de sistema. Se existe somente um retorno possível, então esta não é uma informação devolvida do sistema, mas uma confirmação da transação finalizada. Este tipo de passo também não possui praticamente influência no esforço

para desenvolver um sistema, porque eles são parte da interface relacionada ao evento de sistema já implementado (WAZLAWICK, 2004). Em alguns casos, um passo de um caso de uso pode ter tanto uma entrada quanto uma saída de sistema. Neste caso ele é marcado com [EV] [RS] e são contados dois passos obrigatórios.

A Figura 2 apresenta um exemplo de um caso de uso "Emprestar Fitas" de um sistema de Videolocadora, o mesmo apresentado na Figura 1, onde os passos obrigatórios são identificados.

Fluxo Principal	Fluxos Alternativos
1. O cliente chega ao balcão com as fitas que deseja locar. 2. O cliente informa seu nome e entrega as fitas ao funcionário. 3. [EV] O funcionário registra o nome do cliente e inicia a locação. 4. [EV] O funcionário registra cada uma das fitas. 5. [RS] O funcionário finaliza a locação, devolve as fitas ao cliente e lhe informa a da data de devolução e o valor total da locação. 6. O cliente deixa a locadora com as fitas.	**3a. O cliente não possui cadastro** 3a.1 O cliente deve informar seus dados para cadastro. 3a.2 [EV] O funcionário registra o cadastro. 3a.3 Retorna ao Fluxo Principal no passo 3. **3b. O cliente possui pendências no cadastro (locação anterior não foi paga)** 3b.1 O cliente paga seu débito. 3b.2 [EV] O funcionário registra o pagamento, eliminando a pendência. 3b.3 Retorna ao Fluxo Principal no passo 3. **4a. A fita está reservada para outro cliente** 4a.1 [RS] O funcionário informa que a fita não está disponível para locação. 4a.2 Prossegue a locação no Fluxo Principal no passo 4 sem incluir a fita reservada. **4b. A fita está danificada** 4b.1 O funcionário informa que a fita está danificada. 4b.2 [EV] O funcionário registra que a fita está danificada. 4b.3 O funcionário verifica se existe outra fita disponível com o mesmo filme. 4b.3 Se existir, o funcionário substitui a fita e Retorna ao Fluxo Principal no passo 4, sem registrar a fita danificada.

FIGURA 2 - CASO DE USO: SISTEMA VIDEOLOCADORA (WAZLAWICK, 2004)

Além disso, a fim de produzir descrições úteis de caso de uso, o estilo essencial deve ser usado (LARMAN, 2002), como será descrito melhor no próximo tópico.

2.1.4 Descrição Essencial e Real do Caso de Uso

A descrição essencial é descrita por COCKBURN (2005) como “deixe de fora a interface do usuário; focalize a intenção”. WAZLAWICK (2004, p.63) apresenta como “o que” acontece entre o usuário e o sistema e não “como” isto ocorre.

Na fase de análise todos os casos de uso devem ser descritos com o estilo essencial, ou seja, somente a “essência” das operações é apresentada. Na fase de projeto os casos de uso são descritos em estilo real, ou seja, “como” realmente irá acontecer, sua realização concreta (WAZLAWICK, 2004, p.63-67).

Na descrição essencial deve-se evitar descrever os processos internos, interface e a tecnologia utilizada no sistema. Passos como “O funcionário clica no botão ‘procurar’ digitando o código do cliente no campo” ou “O cliente passa o cartão magnético” são considerados impróprios na fase de análise. Ao invés disso, deve-se descrever a operação no estilo essencial como “O funcionário localiza as informações do cliente” ou “O cliente se identifica”. Este estilo é sugerido quando o analista não conhece o sistema, porque diversas alternativas para implementar as operações e as interfaces poderão surgir na fase de projeto (WAZLAWICK, 2004, p.63-67).

Já na fase de projeto, são definidas as versões reais dos casos de uso, propondo uma solução informatizada e com as tecnologias a serem usadas. Este estilo é sugerido quando o sistema é bem conhecido e as interfaces estão todas especificadas como na implementação de um sistema já existente, mas que será implementado com uma nova tecnologia. Neste caso não é necessário descrever os casos de uso no estilo essencial, mas sim descrevê-los diretamente no estilo real (WAZLAWICK, 2004, p.63-64).

2.1.5 Escrevendo Casos de Uso para o propósito de estimativa

Conforme ANDA et al. (2001), os casos de uso são partes-chave no processo de contagem do método PCU e devem ser escritos em um nível adequado de detalhes. Deve ser possível contar as transações nas descrições dos casos de uso a fim de definir a complexidade do caso de uso. O nível de detalhe nas descrições e a estrutura do caso de uso têm grande impacto na precisão das estimativas baseadas em casos de uso. O

Modelo de Caso de Uso pode também conter um variado número de atores e casos de uso e estes números também afetarão a precisão das estimativas.

RIBU (2001) propôs um princípio básico segundo o qual um caso de uso que tenha mais de dez etapas no cenário principal de sucesso deve ser reduzido, unindo-se as etapas ou separando as funcionalidades. Uma razão importante para que o caso de uso não seja muito longo é que a escala de complexidade no método Pontos de Caso de Uso tem somente três níveis: simples, médio e complexo, não sendo assim capaz de diferenciar a complexidade dos casos de uso com muitas transações.

Em ANDA et al. (2001) foi relatado que haveria a necessidade de uma escala de complexidade com mais níveis para modelar a funcionalidade de um projeto.

WAZLAWICK (2004) propôs a classificação dos passos obrigatórios. Utilizando esta classificação e contando somente os passos obrigatórios ao invés de todos os passos dos casos de uso, pode ser esperada uma menor variação na estimativa feita por diferentes analistas para o mesmo sistema, conforme mostrado por VIEIRA & WAZLAWICK (2006). Espera-se que todos os analistas descrevam a mesma quantidade de passos obrigatórios, mas não os passos complementares que não tem nenhuma influência no desenvolvimento de software.

2.2 Métricas de Estimativas de Software

Estimativas são exigidas quando é preciso planejar e gerenciar projetos de software (PRESSMAN, 1995). Em muitos casos, as estimativas são feitas a partir de bases históricas e experiências de especialistas, mas também existem modelos de estimação formais para serem utilizados.

Exemplos desses modelos são Linha de Código, Análise de Pontos de Função e mais recentemente Pontos de Casos de Uso que serão apresentados a seguir.

2.2.1 Linhas de Código

Umas das primeiras métricas de software, Linhas de Código (LOC) surgiu no tempo em que as pessoas utilizavam linguagens como Fortran e Assembly, e utilizava-

se cartão perfurado para entrada de dados. Um cartão perfurado geralmente representava uma linha de código (WIKIPEDIA, 2006).

Estudantes da área de Computação geralmente iniciam as estimativas de software com LOC por ser mais intuitiva. Fazem comparações entre si em quantidade de linhas de código que utilizaram para desenvolver um mesmo problema específico (VAZQUEZ, 2003). Porém, este método possui várias desvantagens:

a) A inclusão na contagem da quantidade de linhas comentadas, linhas em branco, etc.

b) Não existe um padrão para a contagem em LOC.

c) Esta medida não tem significado para os clientes, só para programadores.

LOC foi uma métrica bastante utilizada até meados de 1970. Logo surgiram diversas linguagens de programação nas quais o conceito de linhas de código não fazia sentido, e conseqüentemente a necessidade de outras formas de estimar o tamanho de software (ANDRADE, 2004, p.22).

2.2.2 Análise de Pontos de Função

A Análise de Pontos de Função (APF) surgiu no início da década de 70, sendo proposta por A. J. Albrecht da IBM, como um meio de medir o tamanho de software e a produtividade das equipes de desenvolvimento. Esta foi uma alternativa às métricas com base em linhas de código. A técnica foi apresentada à comunidade no final da década de 70. Após sucessivos trabalhos de Capers Jones demonstrando sua importância, houve um crescimento dos usuários de APF. Também foi importante a fundação em 1986 do *International Function Point User Group* (IFPUG), responsável pela definição das regras de contagem, pelo treinamento e pela certificação dos profissionais interessados na aplicação dessa métrica e divulgação de diversas bases de dados históricas, disponibilizadas por vários órgãos, entre eles o International Software Benchmarking Standards Group (ISBSG). Em 1990, foi lançada a primeira versão do CPM – *Counting Practices Manual* ou Manual de Práticas de Contagem com o objetivo de padronização da técnica. No final de 2002 a APF, conforme definida na versão 4.1 do manual do

IFPUG, passou à condição de padrão internacional, através da norma ISO/IEC 20926:2002 (VAZQUEZ, 2003, p. 39-43).

A APF mede o desenvolvimento do software do ponto de vista do usuário pela quantificação das funcionalidades fornecidas. A base do processo de contagem é focada na especificação dos requisitos, que vão mapear as necessidades dos usuários para as características e funcionalidades do software, as quais poderão ser medidas ou contadas (VAZQUEZ, 2003, p. 48). Este foi o primeiro método para dimensionamento de software independente de tecnologia, o que possibilita comparar o desempenho de projetos usando diferentes tecnologias e estimar o esforço mais adiantado no ciclo de vida do projeto (RIBU, 2001).

Conforme RIBU (2001), a métrica APF é baseada em cinco atributos externos das aplicações de software:

a) Entradas para a aplicação.

b) Saída da aplicação.

c) Consulta pelos usuários.

d) Arquivos Lógicos ou Arquivos de Dados para serem atualizados pela aplicação.

e) Interfaces para outras aplicações.

Estes componentes são pesados por complexidade e adicionados ao valor "Pontos de Função Não-Ajustados"[1]. Albrecht não dá nenhuma justificativa para os valores usados no sistema de pesagem, exceto que eles dão "bons resultados" e que eles foram determinados por "debates e testes" (RIBU, 2001). Após a identificação e classificação de todos esses componentes, o número de pontos de função não-ajustado é multiplicado pelo fator de ajuste. O fator de ajuste tornou-se opcional para se adequar ao padrão ISO/IEC de medição funcional. A razão do fator de ajuste se dá pelo fato de ajustar o valor de Pontos de Função Não-Ajustado conforme 14 CGs - Categorias Gerais, que

[1] *Unadjusted Function Points*

integram alguns requisitos não-funcionais. O resultado disto gera o "Pontos de Função Ajustados"[2], que podem ser de três tipos de contagem: projeto de desenvolvimento, projeto de melhoria ou projeto de aplicação (VAZQUEZ, 2003, p. 61-62).

O método original de Pontos de Função tem sido modificado e refinado por diversos grupos, mas o método tem sérias limitações. Uma destas é que a Análise de Pontos de Função foi desenvolvida para aplicações de processamento de dados. Seu uso em aplicações científicas e de tempo-real é controverso (FENTON & PLEEGER, 1997). Além disso, desenvolvimento Orientado a Objetos e a Análise de Pontos de Função não têm chegado a um acordo (SPARKS & KASPCZYNSKI, 1999). Conceitos como objetos, casos de uso e interface gráfica do usuário (GUI – Graphical User Interface) não podem ser traduzidos em vinte anos de conceitos de "entradas elementares" e "arquivos lógicos" (RULE, 2001).

Vários estudos têm sido realizados propondo melhorias na Análise de Pontos de Função, conforme apresentadas a seguir.

2.2.2.1 Trabalhos relacionados à Análise de Pontos de Função

Em relação a software OO, vários estudos têm sido realizados sobre o uso e mapeamento dos conceitos da APF para conceitos OO. ANDRADE (2004, p. 19-22) relata:

> "Alguns pesquisadores propuseram o mapeamento da APF para OO com base no modelo de casos de uso da Object Oriented Software Engineering (OOSE) (FETCKE; ABRAN; NGUYEN, 1997). Outros autores basearam-se no diagrama de classes (consideram classes como arquivo lógico e as mensagens como transações) (WHITMIRE e ASMA, ambos citados por FETCKE; ABRAN; NGUYEN (1997); CALDIERA et al. (1998)). Uemura; Kusumoto; Inoue (1999) detalharam as regras de medição da APF para a especificação de requisitos e projetos usando os diagramas de seqüência e de classes com base na Unified Modelling Language (UML) (KUSUMOTO; INOUE; KASIMOTO, 2000). Por outro lado, Gupta R. e Gupta S. (1996) acreditam que não existe um simples relacionamento entre os conceitos originais da APF e os conceitos de objetos e serviços."

---

[2] A*djusted Function Points*

Outros autores dedicaram a novos métodos como Fast Count, Object Oriented Function Point (OOFP), Predictive Object Point (POPs), Object Oriented Design Function Points (OODFP), Fast&&Serious e Use Case Points (UCP), desenvolvido por Arnold e Pedross (1998), com o mesmo nome para o método proposto por Karner (1993) e é baseado em cenários e casos de uso.

> "De acordo com a comparação dos resultados da contagem dos PCU (proposto pelo método de Arnold e Pedross, 1998) e dos FP, eles detectaram uma variação na estimativa entre 90% a 110% e ressaltaram que "esta precisão é suficiente porque a avaliação da métrica de APF pode ter uma imprecisão de até 30%, quando a estimativa do tamanho é realizada por diferentes líderes de projetos." (ANDRADE, 2004, p. 19-22)

ANDRADE (2004, p. 19-22) ainda relata que:

> "Além dos diversos estudos apresentados sobre a aplicação da APF em projetos OO, um outro estudo relevante foi realizado em 1994 pelo *Quality Assurance Institute* e IFPUG. Este estudo indica que a variação da contagem entre contadores treinados e certificados tem diminuído para aproximadamente 11%. No entanto, outras pesquisas realizadas por Uemura; Kusumoto; Inoue (1999), Arnold e Pedross (1998) e Kemerer; Low; Jeffery estes últimos citados por Kitchenham (1997), ressaltam que há diferença de 10% a 12% entre os contadores de uma mesma organização e de 30% entre contadores de organizações diferentes."

A pesquisa de KITCHENHAM & KÄNSÄLÄ (1997) mostrou que o fator de ajuste não melhora a estimativa da Análise de Pontos de Função, tanto é que o IFPUG tornou-o opcional para adequar-se ao padrão ISO/IEC de medição funcional[3].

2.2.3 Análise de Pontos de Função Mk II

A Análise de Pontos de Função MkII ou Mark II é uma variação da Análise de Pontos de Função usada principalmente no Reino Unido. Este método foi proposto por

---

[3] "A norma ISO/IEC 14143 foi desenvolvida para garantir que todos os métodos de Medição Funcional de Tamanho sejam baseados em conceitos similares e que possam ser testados para assegurar que eles se comportam de maneira similar e da forma esperada por um método de medição, dependendo dos domínios funcionais a que se aplicam.Ao final do ano de 2002 a técnica de análise de pontos de função, conforme definida na versão 4.x do manual do IFPUG, foi aprovada como um método de Medição Funcional de Tamanho aderente à norma ISO/IEC 14143." (FATTOCS, 2007)

Charles SYMONS para capturar melhor a contagem de complexidade de processamento interno (SYMONS, 1991). SYMONS observou alguns problemas com a contagem de Pontos de Função original, dentre estes problemas que a escolha dos valores Simples, Médio e Complexo é simplificada demais. RIBU (2001) afirma que muitos itens Complexos não são propriamente pesados.

Diferente da APF original, o MkII vê o sistema como uma coleção de transações lógicas. Cada transação consiste de um componente de entrada, processamento e saída (SYMONS, 1991). Conforme RIBU (2001), um tipo de transação lógica é definido como uma única combinação de entrada/processamento/saída disparada por um único evento de interesse do usuário, por exemplo:

a) Criar um cliente.

b) Atualizar uma conta.

c) Consultar em uma ordem de estado.

d) Produzir um relatório.

RIBU (2001) ainda fez a observação que a definição de transações lógicas é muito similar ao conceito de um caso de uso. Por causa destas similaridades entre os conceitos, as duas abordagens podem ser combinadas para produzir estimativas melhores sob certas circunstâncias.

O método Mk II estende a lista de Fatores de Ajuste Técnico da Análise de Pontos de Função. Estes fatores foram descartados porque foi decidido que eles não fazem sentido no desenvolvimento de software atual. Entretanto, muitos profissionais têm ignorado o ajuste e trabalhado usando os Pontos de Função Não-Ajustados (RULE, 2001).

A lista das características (Quadro 1) é apresentada a fim de mostrar que KARNER (1993) adaptou os fatores T15, T16, T17 e T18 para usar no método Pontos de Caso de Uso, que será apresentado a seguir, além dos fatores T1 a T14 propostos por Albrecht (RIBU, 2001).

QUADRO 1 - FATORES DE AJUSTE TÉCNICO

T1 Comunicação de Dados	T11 Fácil Instalação
T2 Funções Distribuídas	T12 Fácil Operação
T3 Performance	T13 Múltiplos Locais
T4 Configuração altamente utilizada	T14 Modificação Facilitada
T5 Volume de Transação	T15 Requisitos de Outras Aplicações
T6 Entrada de Dados On-line	T16 Segurança, Privacidade, Auditoria.
T7 Eficiência do Usuário Final	T17 Necessidade de Treinamento do Usuário
T8 Atualizações On-line	T18 Uso Direto por Terceiros
T9 Processamento Complexo	T19 Documentação
T10 Usado em outras Aplicações	T20 Características Definidas pelo Cliente

Fonte:Adaptado (ANDRADE, 2004; KARNER, 1993; RIBU, 2001).

Esta lista pode ser comparada com a lista de fatores técnicos do método Pontos de Caso de Uso que serão mostrados mais adiante, a fim de detectar que muitos destes fatores são idênticos.

O método Mk II, como o método Pontos de Função original, não é tipicamente útil para estimar projetos de software que incluem muitos cálculos em tempo-real ou embutidos, projetos que tem uma quantia substancial de fundamento algorítmico ou projetos web (RULE, 2001).

2.2.4 Pontos de Caso de Uso

Gustav Karner desenvolveu em 1993 o método Pontos de Caso de Uso para estimar o desenvolvimento de software orientado a objetos. O método PCU tem como base a análise de Pontos de Função e o método MkII, uma adaptação da análise de Pontos de Função (KARNER, 1993; RIBU, 2001).

O método PCU fornece uma estimativa de software logo na fase inicial, tendo como entrada os casos de uso. Após os casos de uso serem escritos, é possível gerar uma estimativa com a contagem PCU – este é o nome do método e a unidade de medida do método (ANDRADE, 2004; RIBU, 2001).

RIBU (2001) e outros relatam que o método não é novo, mas não tem se tornado popular embora possua algumas vantagens como:

a) Facilidade de uso.

b) Processo de contagem simples.

c) Estimativas podem ser rapidamente calculadas com uma planilha eletrônica.

d) Produz estimativa na fase de Levantamento de Requisitos no processo de desenvolvimento de software.

Uma das razões para que isto esteja ocorrendo deve-se ao fato que:

a) Apesar de casos de uso serem bastante utilizados para descrever requisitos funcionais, ainda não existem padrões para escrever casos de uso e não existem bons históricos de produtividade.

b) O método não foi incorporado em ferramentas populares de estimativa de software.

c) A Análise de Pontos de Função tem se tornado padrão internacional e é a métrica mais usada no mercado e provê estimativas mais precisas à medida que se obtém mais informações do projeto.

A seguir é apresentado o processo de contagem do método Pontos de Caso de Uso originalmente proposto por KARNER (1993).

2.2.4.1 Contagem dos Pontos de Caso de Uso

Quando todos os casos de uso já estiverem em um nível adequado de detalhamento, segue o processo de contagem:

1. Os atores são classificados conforme as categorias do Quadro 2. O total de UAW (Unadjusted Actor Weight) é calculado pela contagem do total de atores em cada categoria, multiplicando cada total pelo seu específico fator de peso e adicionando os produtos.

QUADRO 2 - CLASSIFICAÇÃO DE ATOR

Ator	Peso	Descrição
Simples	1	Representa outro sistema com uma API (Interface para Programação de Aplicações) definida.
Médio	2	Representa outro sistema que interage através de um protocolo como TCP/IP ou FTP.
Complexo	3	Uma pessoa interagindo através de uma interface gráfica ou página da Internet.

Fonte:Adaptado (ANDRADE, 2004; RIBU, 2001).

2. Os casos de uso são classificados conforme o Quadro 3 e dependem do número de transações ou número de classes de análise envolvidas no processo. Uma transação é um evento que ocorre entre um ator e o sistema, o qual é executado completamente ou não. O total de transações de um caso de uso é a soma das transações do fluxo principal com os fluxos alternativos.

A contagem do número de transações pode ser feita contando o número de etapas (passos) do caso de uso (RIBU, 2001; VALENÇA, 2003; BANERJEE, 2001). Os casos de uso incluídos e estendidos não são considerados (ANDA, 2002; RIBU, 2001).

Os UUCW (Unadjusted Use Case Weights) são calculados pela contagem do número de casos de uso em cada categoria, multiplicando cada categoria de caso de uso com seus pesos e adicionando os produtos. O UAW é adicionado ao UUCW para determinar o UUPC (Unadjusted Use Case Points).

QUADRO 3 -CLASSIFICAÇÃO DO CASO DE USO

Caso de Uso	Peso	Descrição
Simples	5	3 ou menos transações. Deverá ser realizado com menos 5 classes de análise.
Médio	10	4 a 7 transações. Deverá ser realizado com 5 a 10 classes de análise.
Complexo	15	Mais que 7 transações. Deverá ser realizado com pelo menos 10 classes de análise.

Fonte:Adaptado (ANDRADE, 2004; RIBU, 2001).

3. Os pontos de caso de uso são ajustados com base nos valores determinados conforme os fatores de complexidade técnica (Quadro 4) e os fatores ambientais (Quadro 5). Os

fatores técnicos correspondem a uma série de requisitos funcionais e os fatores ambientais correspondem aos requisitos não-funcionais, bem como experiência e motivação da equipe (FREIRE, 2003). Em cada fator é determinado um valor entre 0 e 5, dependendo da influência no projeto. Significando: 0-nenhuma influência (irrelevante), 1-influência mínima, 2-influência moderada, 3-influência média, 4-influência significativa e 5-grande influência (essencial).

QUADRO 4 -FATORES DE COMPLEXIDADE TÉCNICA

Fator	Descrição	Peso
T1	Sistema Distribuído	2
T2	Desempenho da aplicação (Performance)	1
T3	Eficiência do usuário final	1
T4	Processamento interno complexo	1
T5	Reusabilidade (código reusável)	1
T6	Facilidade de instalação	0,5
T7	Usabilidade (fácil para utilizar)	0,5
T8	Portabilidade	2
T9	Facilidade de manutenção	1
T10	Concorrência	1
T11	Características especiais de segurança	1
T12	Acesso fornecido a terceiros	1
T13	Facilidades especiais de treinamento dos usuários	1

Fonte: (ANDRADE, 2004; RIBU, 2001).

O TCF (Technical Complexity Factor) é calculado pela multiplicação da influência pelo seu peso, adicionando todos estes números para dar à soma chamada TFactor. Aplica-se então a seguinte fórmula: TCF = 0.6 + (0.01 * TFactor).

QUADRO 5 - FATORES AMBIENTAIS

Fator	Descrição	Peso
F1	Familiaridade com o Processo Unificado (RUP)	1,5
F2	Experiência na Aplicação	0,5
F3	Experiência com Orientação a Objetos	1
F4	Capacidade do líder do projeto	0,5
F5	Motivação	1
F6	Requisitos estáveis	2
F7	Trabalhadores com dedicação parcial	-1
F8	Dificuldade na linguagem de programação	-1

Fonte:(ANDRADE, 2004; RIBU, 2001).

O EF (Environmental Factor) é calculado de acordo com a multiplicação da influência pelo seu peso, adicionando todos os valores à soma chamada EFactor. Aplica-se então a seguinte fórmula: EF= 1.4 + (0.03 * EFactor). Nos dois casos, fatores técnicos e fatores ambientais, se todas as influências forem iguais a 3, então estes fatores não influenciarão no resultado da estimativa, pois nos dois casos as fórmulas terão aproximadamente o valor igual a 1.

4. Os PCU (pontos de caso de uso) são calculados com a seguinte fórmula:

   PCU = UUCP * TCF * EF.

2.2.4.2 Produtividade com o método Pontos de Caso de Uso

Quando KARNER (1993) apresentou o método à sociedade, o método foi aplicado a três projetos da Objectory, a fim de propor uma técnica de estimação para o processo Objectory de Ivar Jacobson. Os resultados foram simplesmente tirados da experiência da equipe de desenvolvimento, o que gerou a média de 20 homens/hora por PCU e os valores de ajuste. Conforme KARNER (1993), “mais dados de projetos deveriam ser reunidos para ajustar o modelo, pesos e constantes”. Outras experiências de campo demonstraram variações entre 15 e 30 homens/hora (SPARKS & KASPCZYNSKI, 1999).

SCHNEIDER & WINTER (1998) sugeriram um refinamento da contagem de KARNER (1993) para a quantidade de homens/hora, determinada pelos fatores ambientais. O número dos fatores entre F1 e F6 que são abaixo de 3, são contados e adicionados ao número de fatores entre F7 e F8 que são acima de 3:

a) Se o total for 2 ou menor, então use o valor 20 homens/hora por PCU.

b) Se o total for 3 ou 4, então use o valor 28 homens/hora por PCU.

c) Se o total for acima de 4, é recomendado que sejam feitas mudanças no projeto e o número deve ser ajustado ou em últimos casos, usar o valor 36 homens/hora por PCU.

A razão para utilizar esta abordagem é que os fatores ambientais significam o nível de experiência da equipe e a estabilidade do projeto. Quaisquer valores negativos significam que serão gastas mais horas treinando pessoal ou corrigindo problemas devido à instabilidade, e com isto menos tempo será dedicado ao projeto. Por conta disto à sugestão de usar mais homens/hora por PCU. É preciso estar atento que qualquer ajuste nos fatores ambientais pode implicar em grandes mudanças no resultado final da estimativa. Mesmo meio ponto pode implicar em uma mudança muito significativa SCHNEIDER & WINTER (1998).

RIBU (2001) propôs que o valor de homens/hora deve ser calibrado para a organização específica. Deve ser feita a aplicação do método PCU em projetos piloto e então definir o valor homens/hora para a equipe de desenvolvimento de software da empresa.

2.2.4.3 Contagem dos Casos de Uso Estendidos e Incluídos

KARNER (1993) recomendou que os casos de uso estendidos e incluídos não devem ser contados.

ANDA et al. (2001), descreveram que estes casos de uso devem ser contados se contiverem muitas das funcionalidades essenciais. Uma vez que em determinado projeto foram omitidos os casos de uso estendidos e incluídos e resultou numa estimativa mais próxima ao esforço real, isso foi feito porque contando estes casos de uso, obtiveram uma estimativa muito maior que o esforço real. Em outro projeto, estes casos de uso foram contados porque muitas das funcionalidades do projeto foram descritas nestes casos de uso.

Conforme RIBU (2001), parece difícil formar uma regra precisa para contar casos de uso estendidos ou incluídos, mas de acordo com ANDA et al. (2001) nem sempre se deve seguir as recomendações de KARNER (1993). RIBU (2001) ainda relata que na dúvida, é melhor contar os casos de uso estendidos e incluídos, assegurando que todas as funcionalidades serão contadas, a fim de evitar uma baixa estimativa, prejudicando o projeto.

2.2.4.4 Problemas com a contagem dos casos de uso e o método Pontos de Caso de Uso

Não existe um padrão adotado para descrever e estruturar os casos de uso e muitas variações de estilo pode conduzir a medidas diferentes de complexidade dos casos de uso (SMITH 1999). COCKBURN (2005) encontrou 18 definições diferentes para um mesmo caso de uso e outros praticantes têm relatado encontrar acima de 32 interpretações diferentes. RIBU (2001), ainda relata que na prática, parece ser muita a disparidade em como diferentes pessoas entendem e escrevem os casos de uso, pois como diferentes indivíduos aplicam diferentes regras, mesmo dentro de um projeto, os resultados da aplicação dos casos de uso para medir o software podem ser ambíguos.

SMITH (1999) relatou também que um caso de uso deve descrever o comportamento do ponto de vista do ator, mas isto pode ser muito complexo, especialmente se o sistema tem estados (como a maioria tem). Assim, para descrever este comportamento, pode ser requerido um modelo do sistema que possa levar a uma grande quantidade de níveis de decomposição funcional e detalhes, em uma tentativa de capturar a essência do comportamento. SYMONS (2001), concluiu que um caminho para resolver este problema, seria visualizar as transações lógicas Mk II como um específico caso de uso e que usando esta abordagem levaria aos requisitos no qual são mensuráveis e teria uma grande chance de uma única interpretação. Entretanto, conforme LONGSTREET (2001), muitos profissionais têm procurado converter as medidas de casos de uso em Pontos de Função, porque existe uma extensa experiência com métricas de Pontos de Função.

SYMONS (1991) observou também que na Análise de Pontos de Função a escolha dos valores Simples, Médio e Complexo para classificação foi simplificada demais. Isto significa que muitos itens complexos não são corretamente pesados. O mesmo se aplica ao método Pontos de Caso de Uso.

2.2.4.5 Trabalhos relacionados a Pontos de Caso de Uso

Apesar da pouca utilização do método PCU na indústria, diversos autores já experimentaram o método, computando a estimativa para vários projetos, propondo

formulários padrões para descrever casos de uso, definindo regras para atribuição dos valores aos fatores ambientais e descartando os fatores técnicos. Alguns são apresentados a seguir.

SCHNEIDER & WINTER (1998) sugeriram um refinamento da contagem de KARNER (1993) para a quantidade de homens/hora, determinada pelos fatores ambientais, conforme apresentados anteriormente.

ANDA et al. (2001) observaram que alguns aspectos da estrutura dos modelos de casos de uso tiveram impacto nas estimativas tais como: o uso de generalização entre os atores; a contagem dos casos de uso estendidos e incluídos; o nível de detalhe da descrição dos casos de uso; as dificuldades para atribuição de valores aos fatores técnicos e ambientais e; a escolha da taxa de produtividade por PCU. Eles ainda definiram um formulário padrão para descrever os casos de uso. Os resultados deste estudo confirmaram que o modelo de casos de uso tem um impacto forte na estimativa; que o método pode ser utilizado com sucesso para estimar o esforço de desenvolvimento de software; que o método de KARNER (1993) pode apoiar os especialistas no processo de estimativa baseado na APF; e apesar de KARNER (1993) não recomendar a contagem dos casos de uso estendidos e incluídos, ANDA et al. (2001) acham que os mesmos devem ser incluídos na contagem para evitar estimativa abaixo da realidade, principalmente se as funções descritas nesses casos de uso são essenciais e implementadas.

ANDA (2002) aplicou o método como parte de três cursos de modelagem de casos de uso em uma companhia internacional de TI. Dois cursos foram realizados na Dinamarca e um na Noruega, tendo como participantes 37 gerentes e desenvolvedores de projeto de software experientes. O objetivo foi comparar as estimativas do método com as estimativas realizadas pelos especialistas. Os resultados mostraram que a estimativa realizada utilizando o método PCU foi mais próxima ao tempo real de desenvolvimento do que a realizada pelos especialistas, mas ambas as estimativas produzidas foram razoavelmente precisas com base nas informações disponíveis. Com isso, ANDA (2002) concluiu que o método de pontos de casos de uso pode ser utilizado com sucesso para estimar o esforço de desenvolvimento de software e também que a combinação das estimativas dos especialistas e as estimativas com o método PCU

podem ser beneficiadas quando falta experiência específica com o domínio da aplicação e a tecnologia a ser usada.

RIBU (2001) utilizou o método de KARNER (1993) para estimar o tamanho dos projetos de estudantes e da indústria. Ele ainda testou o PCU com e sem o uso dos fatores técnicos de ajuste, calculou a estimativa de esforço, propôs um formulário padrão para descrição de casos de uso e definiu regras para atribuir valores aos fatores ambientais. Além disso, propôs a análise da complexidade do caso de uso através do diagrama de seqüência e de classes que implementam os casos de uso, quando não houver a descrição dos mesmos. Porém, RIBU (2001) ressaltou que esta alternativa pode levar à contagem de classes de projeto e comprometer o resultado da estimativa. As conclusões desse estudo foram as seguintes: o método de PCU teve baixa variação entre o valor estimado e o real; sugeriu descartar os fatores de ajustes técnicos e manter os fatores ambientais; contar os casos de uso incluídos e estendidos quando eles tiverem grande impacto no desenvolvimento; e utilizar um formulário padrão para descrição de casos de uso.

No Brasil, um número reduzido de empresas privadas de desenvolvimento de software tem adotado o PCU para estimar seus projetos ou para estudos experimentais. Algumas dessas experiências já foram publicadas por FEITOSA & JANSEN (2003), CUNHA & ALMEIDA (2003) e VALENÇA (2003).

2.2.4.6 Outros trabalhos

Diferentes métodos foram propostos para medir projetos de software orientados a objetos e computar estimativas de esforço. Alguns métodos são apresentados a seguir.

THOMAS FETKE et al. (1997), propôs um método para mapear abordagens orientadas a objetos em Análise de Pontos de Função, mapeando os casos de uso diretamente no modelo APF usando um grupo de regras concisas que suportam o processo de medição. Uma vez que o conceito de atores no modelo de casos de uso é mais amplo do que o conceito de usuários e aplicações externas em APF, isto não pode ser um-para-um mapeando atores e usuários a aplicações externas. Da mesma forma, todas as aplicações que se comunicam com o sistema sob consideração devem também

aparecer como atores. Isto corresponde ao método PCU proposto por KARNER (1993). O nível de detalhes no modelo de caso de uso pode variar e não fornecer informação suficiente para contar um caso de uso específico de acordo com as regras da APF. Entretanto, como no método PCU, os casos de uso devem ser descritos em mais detalhes a fim de contar as transações. Os autores concluíram que a Análise de Pontos de Função pode ser usada no paradigma OO, pois o método é independente de tecnologia. Entretanto, o método não parece ter sido usado em projetos de desenvolvimento de software com exceção dos projetos descritos neste artigo. Em referência a este trabalho, SMITH (1999) declarou que o nível do caso de uso deve ser descrito apropriadamente para o mapeamento ser válido.

SMITH (1999) da Rational apresentou um framework para estimação baseado em casos de uso, traduzido em Linhas de Código (LOC). Não foi feita mais nenhuma pesquisa neste método, embora a ferramenta "Estimate Professional" que é fornecida pela *Software Productivity Center Inc.* e a ferramenta "CostXpert" da Marotz Inc. produzirem estimativas de esforço por caso de uso, calculando o número de Linhas de Código.

LONGSTREET (2001), da Software Metrics, observou que aplicar Pontos de Função ajuda a determinar se o caso de uso está escrito em um nível adequado de detalhes. Se for possível descrever como os dados passam do ator para dentro da fronteira ou como os fluxos de dados de dentro da fronteira de aplicação passam para o ator, então quer dizer que o nível de detalhes está correto, fora isso o caso de uso necessita de mais detalhes. Adotando ambos os métodos Pontos de Caso de Uso e Pontos de Função, a qualidade do documento de requisitos pode ser melhorada. Dessa forma, a medição e estimativa são melhoradas.

Outros trabalhos como o de ANDRADE (2004), fizeram a combinação dos métodos Pontos de Caso de Uso com Pontos de Função. "Este trabalho teve como objetivo propor aos gerentes uma melhor forma de estimar, revisar e recalcular o tamanho do projeto à medida que novos artefatos estejam disponíveis durante o processo de desenvolvimento do projeto de software orientado a objetos" (ANDRADE & OLIVEIRA, 2004). Os autores estudaram as principais características da APF e PCU e padronizaram os procedimentos de contagem da APF e PCU para projetos de software

OO. Foram realizadas as aplicações em 19 projetos da academia e indústria, chegando à relação entre as APF e PCU, ou seja, com os dados do PCU no início do projeto transformados em APF. Os autores também constataram que existe diferença nas relações da academia e da indústria. Nos projetos da academia o resultado foi a equação linear APF= 1,4615 PCU, enquanto que na indústria foi igual a APF = 2,89 PCU.

3 Teoria para Estimativa de Software Baseada em Casos de Uso no Desenvolvimento Orientado a Objetos

Esta teoria foi criada a partir do desenvolvimento de alguns projetos, aplicando-se o método de desenvolvimento de software orientado a objetos de LARMAN (2002) adaptado por WAZLAWICK (2004).

Após o desenvolvimento dos projetos foram identificados alguns padrões de comportamento entre os artefatos no método utilizado, o que possibilitou definir esta teoria para gerar estimativas de tempo de desenvolvimento de softwares orientado a objetos, assim como é possível também através de métodos já propostos como Análise de Pontos de Função (APF) e Pontos de Casos de Uso (PCU).

O método APF possui várias limitações, destacando-se o fato de ter sido originalmente proposto para o desenvolvimento de softwares estruturado. O método PCU vem sendo amplamente aperfeiçoado, porém herda algumas limitações por ter sido uma evolução do método APF, como por exemplo, a utilização dos Fatores de Complexidade Técnica. Além disso, diversos autores relatam que a classificação dos atores e casos de uso em Simples, Médio e Complexo é muito limitada.

A base para a teoria proposta são os casos de uso, conforme também é utilizado no método PCU, o que difere essencialmente é que o método PCU utiliza todos os passos dos casos de uso para classificá-los, enquanto que a teoria proposta determina que a partir da expansão dos casos de uso existe um determinado conjunto de passos que podem ser diferenciados em “obrigatórios”, enquanto os demais são “complementares”. Os passos “obrigatórios” são aqueles que determinam o esforço de desenvolvimento, proporcionalmente ao seu número, enquanto que os passos “complementares” não têm influência sobre o desenvolvimento (WAZLAWICK, 2004).

Após a classificação dos casos de uso, no método PCU deve-se multiplicar os pesos dos casos de uso aos fatores ambientais para estimar o esforço necessário para o desenvolvimento de software, enquanto que esta teoria propõe que os fatores ambientais sejam não apenas multiplicados, mas somados à complexidade dos casos de uso,

dependendo de sua natureza. Por exemplo, a necessidade de estudar uma determinada ferramenta de desenvolvimento ou a criação de um determinado *framework* que vai acelerar etapas posteriores do desenvolvimento, usualmente não depende da complexidade do sistema a ser desenvolvido. Mais do que isso, essas atividades podem causar aumento do esforço, especificado por uma constante, que independe da complexidade final do sistema e uma diminuição no esforço, determinada por uma constante *dependente* da complexidade do sistema.

A teoria é composta por um conjunto de equações que são aplicadas em diferentes momentos da fase de análise do processo de desenvolvimento de software orientado a objetos. Na medida em que os artefatos vão sendo gerados, uma equação específica determinará a estimativa conforme os dados disponíveis. Quanto mais artefatos forem gerados na fase de análise, mais precisa será a estimativa. A teoria e suas equações procuram explicar *porque* a contagem de passos obrigatórios gera estimativas mais próximas do esforço real do que a simples contagem de passos ou intuição de analista. Isso diferencia a teoria aqui apresentada dos demais métodos de contagem de pontos de casos de uso, pois estes últimos foram adaptados do método de pontos de função, idealizado inicialmente para a análise estruturada. A teoria aqui apresentada mostra o esforço despendido que cada passo obrigatório pode gerar no processo de análise orientada a objetos.

A seguir, serão apresentadas as etapas relevantes e as definições da teoria proposta para calcular as estimativas.

3.1 Primeira Etapa: Listagem dos Casos de Uso na Fase de Concepção

Vários processos da análise (ex.: estudo de viabilidade, levantamento de requisitos) ocorreram antes de os casos de uso serem levantados na fase de concepção [LARMAN, 2002], mas estes processos não serão detalhados aqui, visto que o esforço de desenvolvimento é calculado a partir dos casos de uso. Nesta etapa são listados os casos de uso que compõe o sistema, e uma breve descrição de cada um deles é

apresentada, sendo que a experiência do analista é determinante para estimar a complexidade de cada caso de uso.

Com esta informação, pode-se trabalhar com dois tipos de estimativa:

a) A que considera uma complexidade média inerente aos casos de uso.

b) A que utiliza a experiência do analista para classificar os casos de uso conforme sua complexidade que pode ser discreta, por exemplo, Simples=1, Médio=3 e Complexo=5; ou contínua, onde o caso de uso mais simples terá o valor próximo de 0(zero), o médio com valor próximo de 1 e o complexo próximo de um valor limite positivo que pode ou não ser estabelecido para efeito de medida.

Para efeito desta teoria será utilizada a escala contínua, com o caso de uso simples valendo próximo a 0, o médio valendo 1 e o complexo valendo qualquer valor maior do que 1(ilimitado). Em fases subseqüentes da análise, ao invés da intuição do analista, poderão ser usadas informações mais precisas sobre a complexidade de cada caso de uso.

3.1.1 Estimativa usando apenas o número de casos de uso

A primeira técnica de estimativa possível utilizando a listagem dos casos de uso, pode inicialmente considerar dois fatores:

a) O número de casos de uso: N .

b) O tempo médio para desenvolver todos os artefatos (inclusive o código) que derivam de um caso de uso: T .

O tempo médio para desenvolver todos os artefatos derivados de um caso de uso, pode ser calculado experimentalmente a partir de vários projetos, tomando-se o tempo total para desenvolver cada projeto (TD) dividido pelo número de casos de uso (N). Então, o tempo para desenvolver um caso de uso (T) será calculado como a média dos valores TD / N em vários projetos.

A equação de estimativa de tempo pode ser, portanto, nesta fase, enunciada assim:

(1) $TDE = N * T$

onde TDE é o Tempo total de Desenvolvimento Estimado para o software.

Esta técnica de estimativa tende a ser pouco precisa, pois a simples média não dará bons resultados quando a complexidade dos casos de uso for variável, ou seja, espera-se que pelas características distintas dos projetos e diversidade de tipos de casos de uso, o desvio padrão seja muito alto.

Sendo assim, é necessário adicionar a esta equação uma medida de complexidade para cada caso de uso, o que será apresentado na subseção seguinte.

3.1.2 Acrescentando uma medida de complexidade aos casos de uso

A segunda técnica considera uma medida subjetiva do analista para classificar a complexidade dos casos de uso.

Esta técnica poderá ser mais precisa que a anterior à medida que o analista seja capaz de apresentar uma estimativa de complexidade mais próxima da realidade para cada caso de uso (ANDA, 2002). Esta técnica aproxima-se mais do método PCU, que classifica os casos de uso em Simples, Médio e Complexo.

A variável T continua denotando o tempo médio para desenvolver um caso de uso, mas ela é agora multiplicada pelo fator de complexidade específico para cada caso de uso, definido pelo gerente de projeto. Este fator é dado por uma função $complex(uc)$, a qual associa um valor racional entre 0 e infinito, a cada caso de uso (uc).

A equação de estimativa de esforço pode ser assim enunciada:

(2) $TDE = T * \sum_{i=1}^{N} complex(uc_i)$

Para que a equação 2 seja consistente com a equação 1, deve-se presumir inicialmente que na média $\sum_{i=1}^{N} complex(uc_i) = N$, ou seja, a somatória dos valores de complexidade atribuídos a cada caso de uso em um projeto, deve ser igual ao número de

casos de uso do projeto. Porém, essa situação só ocorre em um projeto ideal. Na prática, os projetos tendem a ter casos de uso mais simples ou mais complexos na média e a fórmula $\sum_{i=1}^{N} complex(uc_i)$ aplicada a um projeto real poderá resultar em um valor diferente de N. Nestes casos, quanto mais negativo for o valor de $\sum_{i=1}^{N} complex(uc_i) - N$, mais simples serão os casos de uso do projeto, e quanto mais positivo for este valor, mais complexos serão os casos de uso do projeto.

3.1.3 Separando fatores dependentes e independentes dos casos de uso.

Se o analista puder estimar o esforço de desenvolvimento que não depende dos casos de uso, então a função $complex(uc)$ pode ser redefinida com mais precisão, como sendo o tempo *efetivamente* empregado no desenvolvimento dos artefatos derivados de um determinado caso de uso.

Seja TFI o tempo total estimado para atividades que independem da quantidade ou complexidade dos casos de uso, então a equação da estimativa de esforço pode ser assim redefinida:

(3) $TDE = TFI + T * \sum_{i=1}^{N} complex(uc_i)$

Exemplos de atividades independentes da complexidade do sistema em desenvolvimento são: treinamento da equipe, desenvolvimento de *frameworks*, aquisição de ferramentas de desenvolvimento, etc. Este valor poderá ser definido pelo gerente de projetos que saberá quais atividades e o tempo que será utilizado para realizar estas atividades.

3.1.4 Considerando que a análise inicial não é completa.

Um aspecto que não pode ser esquecido é que o número de casos de uso N obtido no início da fase análise, tende a ser menor do que o número real de casos de uso

($N' = f * N$) contabilizado quando o sistema estiver pronto, ou seja, o número de casos de uso tende a aumentar à medida que o processo de desenvolvimento prossegue.

Existe um Fator de Aumento (f) do número de casos de uso, que pode nesta etapa inicial ser apenas previsto como uma média. Este fator pode depender do tipo de sistema e também do *estilo de análise*.

Um analista mais despreocupado ou menos experiente, identificará apenas os principais casos de uso quando o número real é muito maior, o que determina um fator f mais alto. Um analista mais detalhista ou mais experiente poderá identificar praticamente todos os casos de uso na fase de concepção, não importando quão elementares eles sejam. Com isso, haverá poucos novos casos de uso a serem descobertos à medida que a análise prossegue. Neste caso, o fator de aumento f terá um valor muito próximo de 1.

Não se trata de considerar aspectos relacionados a novos requisitos, que porventura sejam solicitados pelo cliente durante o processo de desenvolvimento, mas do fator de imprecisão da análise que muitas vezes impede que o analista perceba na fase de concepção, a totalidade dos casos de uso que realmente serão necessários.

Como o valor de f consiste em um valor médio, possivelmente não inteiro, aplicando-se esse fator à equação (3) obtém-se:

$$(4)\ TDE = TFI + T * f * \sum_{i=1}^{N} complex(uc_i)$$

Neste ponto da análise em que apenas a listagem dos casos de uso está disponível, um analista poderá fazer uma previsão de esforço mais precisa à medida que conhecer valores médios históricos para os elementos da equação (4). Não sendo conhecidos alguns destes valores, a fórmula pode ser simplificada para as equações (3), (2) ou (1), perdendo precisão em cada uma delas.

3.2 Segunda Etapa: Expansão dos Casos de uso

A expansão dos casos de uso é gerada a partir dos casos de uso de alto nível identificados na fase de concepção e no documento de requisitos (WAZLAWICK, 2004, p.60). Considerando estas novas informações é possível adicionar mais detalhe à estimativa de esforço para produzir melhores resultados.

Nesta teoria foi utilizado um processo de contagem diferente do método PCU original (KARNER, 1993) que conta todos os passos nos casos de uso. Ao invés disto, foi utilizada a classificação dos passos obrigatórios conforme definida em WAZLAWICK (2004).

De acordo com o método de desenvolvimento de LARMAN (2002), pode-se observar que os diferentes passos descritos nos casos de uso darão origem às transações com o sistema. Cada transação de entrada deve ser modelada e implementada como uma operação de sistema, que opera transformação nos dados. Estas operações possivelmente estarão marcadas com "EV" nos casos de uso, que são ações que levam informações dos atores para o sistema (WAZLAWICK, 2004). Já as transações de saída, devem ser modeladas como consultas, que apenas retornam informações já armazenadas no sistema. Estas consultas estarão marcadas como "RS" nos casos de uso, que são passos que trazem informações do sistema para os atores (WAZLAWICK, 2004). Transações de entrada e saída são consideradas "passos obrigatórios" por WAZLAWICK (2004).

Outras transações poderão não ter nenhuma influência sobre o tempo de desenvolvimento, como as transações entre atores (que não são implementadas) ou transações que descrevem ações do sistema que não implicam na troca de informações como a navegação entre as telas ou exibir a confirmação da realização com sucesso de uma operação de sistema (WAZLAWICK, 2004). Essas transações são consideradas "passos complementares" em WAZLAWICK (2004) e não são utilizadas na contagem dos passos nos casos de uso, ou seja, apenas são contabilizados os passos obrigatórios para efeito do cálculo da complexidade do caso de uso.

Como as transações de entrada e saída devem ser necessariamente implementadas, elas irão gerar impacto na complexidade do caso de uso. Logo, nesta fase de expansão

dos casos de uso pode-se aprimorar a estimativa dos valores $complex(uc)$ para definir o tempo de desenvolvimento de um caso de uso baseado no número de transações obrigatórias (to):

$$(5)\ complex(uc) = f_{uc} \sum_{i=1}^{N_{UC}} complex_{uc}(to_i)$$

Onde: $complex_{uc}(to)$ é uma função que estabelece uma medida de complexidade para uma transação obrigatória, N_{uc} é o número de transações obrigatórias do caso de uso uc e f_{uc} é o fator de aumento esperado no número de transações obrigatórias nas fases posteriores da análise.

Aplicando-se a fórmula 5 em 4, obtém-se uma equação para a estimativa de esforço que considera não só o número de casos de uso, mas também a complexidade individual de cada caso de uso, calculada a partir do número de transações obrigatórias, sua complexidade estimada e o fator de aumento do número destas transações no caso de uso à medida que a análise prossegue:

$$(6)\ TDE = TFI + T * f \sum_{i=1}^{N} \left(f_{uc_i} \sum_{j=1}^{N_{uc_i}} complex_{uc}(to_j) \right)$$

Se f_{uc} for definido como a média histórica, observada do aumento do número de transações obrigatórias dentro de cada caso de uso no momento atual da análise (etapa de expansão dos casos de uso) até a conclusão do sistema, então a equação 6 pode ser simplificada para:

$$(6')\ TDE = TFI + T * f * f_{uc} \sum_{i=1}^{N} \sum_{j=1}^{N_{uc_i}} complex_{uc}(to_j)$$

O tempo estimado para desenvolvimento de um sistema será dado, então, pelo tempo necessário para desenvolver atividades independentes do número de casos de uso do sistema (TFI), somado ao tempo médio para desenvolver todos os artefatos derivados de um caso de uso (T), multiplicado pelo fator de aumento esperado no número de casos de uso ao longo do projeto (f), multiplicado pelo fator de aumento médio no número de passos obrigatórios dentro dos casos de uso (f_{uc}), multiplicado pela complexidade total de passos obrigatórios contabilizados em todos os casos de uso

expandidos ($\sum_{i=1}^{N}\sum_{j=1}^{N_{uc_i}} complex_{uc}(to_j)$). Seja N_{uc} o número total de passos obrigatórios nos casos de uso de um sistema, dado por $N_{uc} = \sum_{i=1}^{N} N_{uc_i}$.

Se todos os passos obrigatórios fossem igualmente complexos, e expressos por um valor médio α, então a equação 6' se reduziria a:

(6'') $TDE = TFI + T * f * f_{uc} * \alpha * N_{uc}$

Ou seja, a complexidade média de um caso de uso seria dada por uma função de 3 variáveis, que são ajustadas através de médias históricas (f , f_{uc}, e α) e o número total de casos de uso do sistema.

3.3 Terceira Etapa: Contratos

Até chegar a esta fase dos contratos, ainda é definido o modelo conceitual que é gerado a partir dos casos de uso expandidos, no qual se tem a troca de informações entre o sistema e o mundo externo.

Nos casos de uso expandidos serão absorvidos os conceitos e atributos, criando o modelo conceitual e é também onde são identificadas as operações e consultas de sistemas (passos obrigatórios) (WAZLAWICK, 2004, p.60-62).

As operações e consultas de sistema são intenções por parte do usuário que são capturadas pelos contratos de operações de sistema e pelos contratos de consulta de sistema (WAZLAWICK, 2004, p.152). Conforme WAZLAWICK (2004, p.152-167) os contratos de operações de sistema podem ser compostos por pré-condições, pós-condições e exceções, enquanto que os contratos para consultas de sistema poderão ter apenas pré-condições e resultados.

As seções que compõem os contratos são definidas assim:

a) **Pré-Condições**: o que necessariamente é verdadeiro quando uma operação ou consulta de sistema for executada. Para que elas sejam verdadeiras, devem existir mecanismos que garantam isso quando o sistema for implementado. Geralmente são mecanismos de interface que impedem que uma operação seja executada antes da

outra, desabilitando botões ou itens de menu relativos àquela operação. Existem ainda dois tipos de pré-condições:

- Garantia de parâmetros: que são aquelas que garantem que os parâmetros sejam elementos válidos do sistema, como por exemplo, que exista um cadastro para o cliente informado como parâmetro para a operação ou consulta.
- Restrição Complementar: que garante que a informação armazenada no sistema está em uma situação determinada, como por exemplo, que o cliente informado não possua débitos em aberto.

b) **Pós-Condições**: as mudanças ocorridas na informação armazenada após a execução da operação de sistema. Somente existem pós-condições nas operações de sistema. Uma operação de sistema não gera resultado – *output*, mas uma mudança interna no estado do sistema.

c) **Exceções**: situações que não podem ser garantidas antes da execução, elas são testadas durante a execução do sistema. Exceções existem somente nas operações de sistema e ocorrem quando não é possível realizar alguma pós-condição, impedindo o funcionamento correto da operação.

d) **Resultados**: ocorrem somente nas consultas que não produzem alteração na informação armazenada, mas apenas apresentam dados que já existem em um formato especificado.

Todas estas informações ajudarão a detalhar ainda mais a estimativa de complexidade individual da operação ou consulta de sistema $complex_{uc}(to)$. Quanto mais pré-condições ou pós-condições e quanto mais exceções ou resultados uma operação ou consulta apresentar, então mais complexa ela será e mais trabalho exigirá do analista para sua correta especificação. Assim, os elementos a serem considerados para calcular a quantidade de trabalho efetivo realizado em cada operação ou consulta de sistema são os seguintes:

n_{par} = número de parâmetros.

n_{pre} = número de pré-condições.

n_{pos} = número de pós-condições.

n_{exc} = número de exceções.

n_{res} = número de resultados.

Cada um destes valores terá alguma influência na complexidade da transação de sistema. Porém, o peso de cada valor poderá ser diferente. É necessário, portanto, não apenas somar os valores, mas obter uma soma ponderada com pesos que devem ser determinados experimentalmente:

$$(7)\ complex_{uc}(to) = p1 * n_{par}(to) + p2 * n_{pre}(to) + p3 * n_{pos}(to) + p4 * n_{exc}(to) + p5 * n_{res}(to)$$

Assim, na fase final da análise, quando os contratos já foram elaborados, a estimação de esforço total pode ser definida a partir da aplicação da equação (7) em (6'):

$$(8)\ TDE = TFI + T * f * f_{uc} \sum_{i=1}^{N} \sum_{j=1}^{N_{uc_i}} \left(p1 * n_{par}(to_j) + p2 * n_{pre}(to_j) + p3 * n_{pos}(to_j) + n4 * n_{res}(to_j) \right)$$

Esta equação consiste, portanto, em uma aproximação da complexidade de um sistema quando sua análise foi concluída. Inicia-se então a fase de projeto.

3.4 A Fase de Projeto e Implementação

Na fase de projeto, segundo LARMAN (2002) e WAZLAWICK (2004), cada transação obrigatória dará origem a um diagrama de comunicação, cuja complexidade dependerá do número de parâmetros, pré-condições, pós-condições e resultados.

No caso das operações de sistema, a complexidade será muito mais dependente do número de pós-condições, porque cada pós-condição corresponderá a uma ação que precisa ser definida no diagrama.

A revisão do modelo conceitual e sua transformação em um diagrama de classes de projeto é um fator independente e pode ser contabilizada desta forma.

O projeto gráfico das interfaces terá forte dependência em relação ao número de parâmetros e resultados das transações obrigatórias.

O projeto do banco de dados será um fator independente assim como a geração do diagrama de classes de projeto, visto que é possível automatizar esta atividade. Mas mesmo não sendo automatizada, ela não depende da complexidade interna das transações, embora possa estar relacionada com o número de casos de uso. O número de classes normalmente está diretamente relacionado ao número de casos de uso.

A fase de implementação pode ser altamente automatizada, caso se disponha de ferramentas adequadas. Este grau de automatização é um fator dependente, pois quanto menos automatizada for a geração de código, maior o tempo gasto com esta atividade. Este tempo é proporcional à complexidade das operações de sistema, especialmente as pós-condições.

4 Avaliação

Neste capítulo são apresentados os estudos de casos utilizados durante o desenvolvimento deste trabalho para comprovar as hipóteses e os objetivos.

4.1 Estudo de Caso 1 – Avaliação da Contagem de Passos

O Estudo de Caso 1 foi realizado com alunos do PPGCC - Programa de Pós-Graduação em Ciência da Computação da UFSC – Universidade Federal de Santa Catarina que na maioria eram funcionários de empresas privadas e já possuíam experiência com o desenvolvimento e estimativas de softwares. Os alunos descreveram casos de uso de dois sistemas fictícios conhecidos (Biblioteca e Hotel). Onze alunos realizaram a tarefa para o Sistema de Biblioteca e oito destes alunos para o Sistema de Hotel.

A primeira etapa foi descrever os casos de uso com os conhecimentos que cada um possuía sobre como descrever um caso de uso e posteriormente aplicada a classificação dos passos conforme sugerido em WAZLAWICK (2004).

A hipótese deste estudo é mostrar que a utilização do processo de contagem de passos do método PCU baseado nos passos obrigatórios sugerido por WAZLAWICK (2004) é menos discrepante do que o processo de contagem do método PCU usando todos os passos, como usualmente é feito.

Devido à falta de padrão na descrição dos casos de uso, diferentes analistas irão descrever passos diferentes para um mesmo caso de uso. A classificação dos passos obrigatórios tende a fazer com que os analistas descrevam a mesma quantidade de passos obrigatórios – ‘EV’ e ‘RS’ – pois sem estes, o caso de uso pode não fazer sentido e posteriormente métodos incompletos poderão ser implementados (WAZLAWICK, 2004).

Abaixo segue um exemplo de caso de uso (Quadro 6) descrito para demonstração, primeiramente um caso de uso com os passos do Fluxo Principal e Fluxos Alternativos do Sistema Biblioteca descritos pelo aluno:

QUADRO 6 - CASO DE USO EMPRESTAR LIVROS

Fluxo Principal	Fluxos Alternativos
1. O cliente chega no balcão com os livros que pretende emprestar. 2. O cliente se identifica e entrega os livros para o funcionário. 3. O funcionário registra o cliente e inicia o empréstimo. 4. O funcionário registra cada livro. 5. O funcionário finaliza o empréstimo, entrega os livros de volta para o cliente avisando a data de devolução. 6. O cliente vai embora com os livros.	**3a. O cliente não possui cadastro** 3a.1 O cliente informa seus dados para cadastro. 3a.2 O funcionário registra o cadastro. 3a.3 Retorna ao Fluxo Principal no passo 3. **3b. O cliente está com sit. pendente (Pgto em aberto)** 3b.1 O cliente efetua o pagamento. 3b.2 O funcionário registra pagamento, eliminando a pendência. 3b.3 Retorna ao Fluxo Principal no passo 3. **4a. O livro está reservado** 4a.1 O funcionário informa que o livro não pode ser emprestado pois possui reserva. 4a.2 Retorna ao Fluxo Principal no passo 4 sem registrar o livro reservado. **4b. O livro está danificado** 4b.1 O funcionário informa que o livro precisará de manutenção e não poderá ser emprestado. 4b.2 O funcionário registra que a fita está danificada. 4b.3 Retorna ao Fluxo Principal no passo 4, sem registrar o livro danificado.

Abaixo, o mesmo caso de uso (Quadro 7) com a classificação dos passos sugerida por WAZLAWICK (2004):

QUADRO 7 - CASO DE USO EMPRESTAR LIVROS (PASSOS OBRIGATÓRIOS)

Fluxo Principal	Fluxos Alternativos
1. O cliente chega no balcão com os livros que pretende emprestar. 2. O cliente se identifica e entrega os livros para o funcionário. 3. [EV] O funcionário registra o cliente e inicia o empréstimo. 4. [EV] O funcionário registra cada livro. 5. [RS] O funcionário finaliza o empréstimo, entrega os livros de volta para o cliente avisando a data de devolução. 6. O cliente vai embora com os livros.	**3a. O cliente não possui cadastro** 3a.1 O cliente informa seus dados para cadastro. 3a.2 [EV] O funcionário registra o cadastro. 3a.3 Retorna ao Fluxo Principal no passo 3. **3b. O cliente está com situação pendente (Pagamento em aberto)** 3b.1 O cliente efetua o pagamento. 3b.2 [EV] O funcionário registra pagamento, eliminando a pendência. 3b.3 Retorna ao Fluxo Principal no passo 3. **4a. O livro está reservado** 4a.1 [RS] O funcionário informa que o livro não pode ser emprestado pois possui reserva. 4a.2 Retorna ao Fluxo Principal no passo 4 sem registrar o livro reservado. **4b. O livro está danificado** 4b.1 O funcionário informa que o livro precisará de manutenção e não poderá ser emprestado. 4b.2 [EV] O funcionário registra que a fita está danificada. 4b.3 Retorna ao Fluxo Principal no passo 4, sem registrar o livro danificado.

No primeiro momento, nota-se que a observação feita por SYMONS (1991) é verdadeira, quando diz que a classificação dos casos de uso foi simplificada demais. Neste caso, se fosse classificar o caso de uso contando os passos no Fluxo Principal e Fluxos alternativos, totalizaria 17 passos, o que no método PCU original faria com que este fosse classificado como Complexo, que é a classificação para casos de uso com a quantidade de passos acima de 7; com isso, 10 passos não seriam contabilizados, sendo que, se estes passos tivessem alguma funcionalidade no sistema, este poderia ser subestimado. Conforme COCKBURN (2005) “bons casos de uso não passam de 9 passos”, se fossem contados somente os Eventos e Respostas de Sistema seriam computados 7 passos, o que no método PCU original faria com que este fosse classificado como Médio.

Os resultados do Estudo de Caso 1 foram colocados no Quadro 6 e 7 para fazer as comparações, gerando a Média e Desvio Padrão (DP). As colunas E1 até E11 no Quadro 8 e E1 até E8 no Quadro 9 correspondem às diferentes equipes de alunos que analisaram os casos de uso:

QUADRO 8 -SISTEMA BIBLIOTECA

Contagem de passos	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**Média**	**DP**
UC 1: Reservar Livro	5	9	6	4	6	6	7	14	8	4	8	7,00	2,83
UC 2: Emprestar Livro	4	8	17	4	16	7	18	18	12	14	14	12,00	5,39
UC 3: Registrar Devolução	5	12	9	8	6	4	11	20	9	13	7	9,45	4,50
UC 4: Registrar Pagamento	6	11	6	4	7	5	-	-	-	-	-	6,50	2,43
TOTAL	**20**	**40**	**38**	**20**	**35**	**22**	**36**	**52**	**29**	**31**	**29**	**32,00**	**9,65**
Contagem EV e RS	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**Média**	**DP**
UC 1: Reservar Livro	3	5	4	2	2	3	4	5	4	2	4	3,45	1,13
UC 2: Emprestar Livro	3	4	7	3	6	3	9	8	6	6	6	5,55	2,07
UC 3: Registrar Devolução	3	5	5	4	3	3	5	9	5	5	4	4,64	1,69
UC 4: Registrar Pagamento	4	5	4	3	4	3	-	-	-	-	-	3,83	0,75
TOTAL	**13**	**19**	**20**	**12**	**15**	**12**	**18**	**22**	**15**	**13**	**14**	**15,73**	**3,47**

QUADRO 9 -SISTEMA HOTEL

Contagem de passos	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**Média**	**DP**
UC 1: Reservar Quarto	9	13	8	8	8	11	6	8	8,88	2,17
UC 2: Cancelar Reserva do Quarto	5	5	3	4	3	3	3	-	3,25	1,58
UC 3: Registrar Serviços Adquiridos	2	10	5	4	-	5	5	7	4,75	3,01
UC 4: Registrar Pagamento	4	7	6	7	9	6	5	5	6,13	1,55
TOTAL	**20**	**35**	**22**	**23**	**20**	**25**	**19**	**20**	**23,00**	**5,24**
Contagem EV e RS	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**Média**	**DP**
UC 1: Reservar Quarto	5	4	4	3	4	5	3	4	4,00	0,76
UC 2: Cancelar Reserva do Quarto	3	2	2	2	2	1	2	-	1,75	0,89
UC 3: Registrar Serviços Adquiridos	2	5	3	3	-	3	3	4	2,88	1,46
UC 4: Registrar Pagamento	2	3	4	5	4	3	3	3	3,38	0,92
TOTAL	**12**	**14**	**13**	**13**	**10**	**12**	**11**	**11**	**12,00**	**1,31**

As figuras 3 e 4 mostram a discrepância da quantidade de passos antes (Quadro 8) e depois (Quadro 9) da classificação dos passos sugerida por WAZLAWICK (2004). As figuras mostram a quantidade de passos mínima, máxima e o desvio padrão de cada caso de uso descrito pelos estudantes.

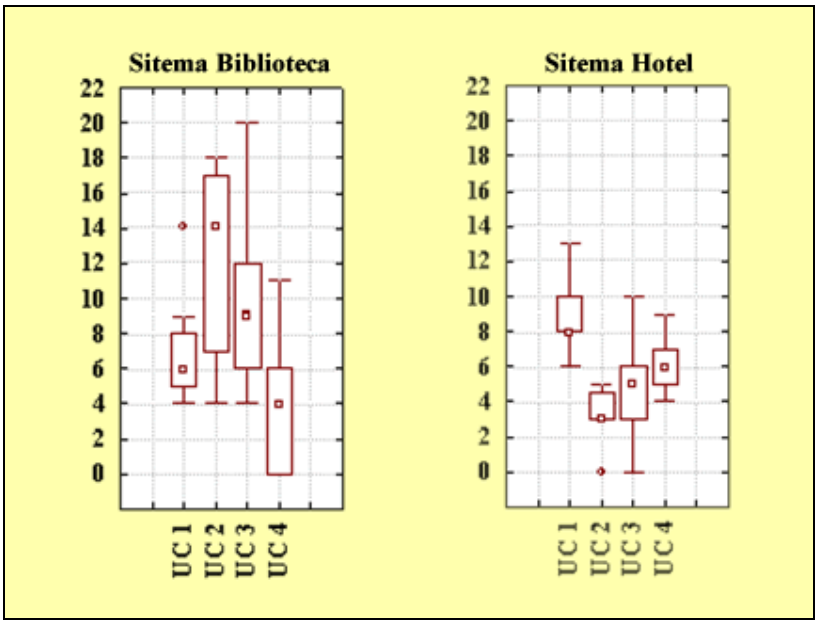

FIGURA 3 - CONTAGEM DE TODOS OS PASSOS

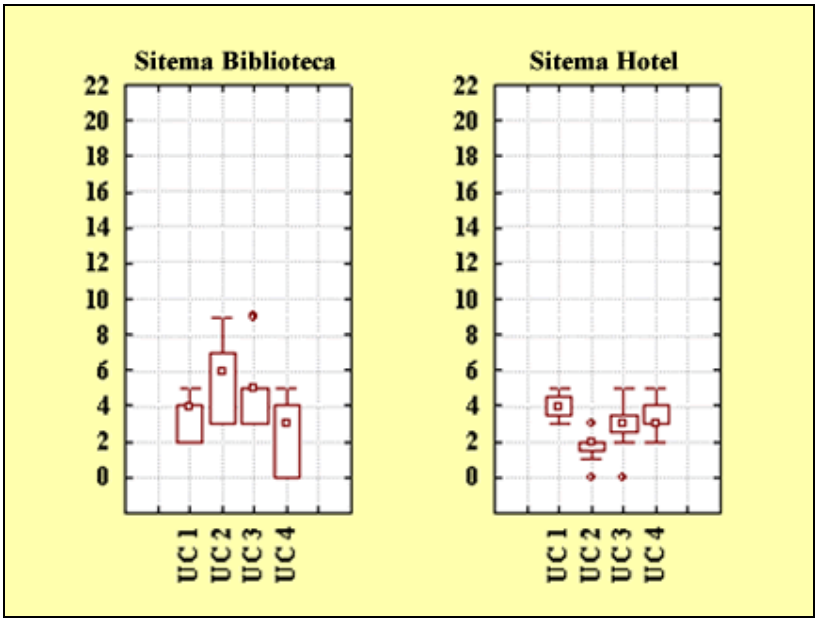

FIGURA 4 - CONTAGEM SOMENTE DOS PASSOS OBRIGATÓRIOS

Com a comparação dos Quadros[4] 8 e 9, foram feitas as seguintes observações:

a) O número de passos mostrados no Quadro 8 é mais disperso que o número de passos obrigatórios mostrado no Quadro 9, conforme pode ser visto nas figuras, pelas caixas retangulares que mostram a discrepância. Por exemplo, de acordo com o Quadro 8, o caso de uso mais discrepante é o UC3 contendo de 4 a 20 passos dependendo do estudante que o escreveu. O mesmo caso de uso tem de 3 a 5 passos obrigatórios conforme o Quadro 9, e é portanto, menos discrepante.

b) Os casos de uso na sua maioria são classificados como Complexo se todos os passos forem contabilizados (Quadro 8), entretanto, a maioria foi classificada como Médio e Simples se somente os passos obrigatórios forem contabilizados (Quadro 9).

c) Contando somente os passos obrigatórios, então os casos de uso chegam à quantidade limite relatada por COCKBURN (2005), que afirma que um caso de uso bem escrito teve ter de 3 a 9 transações ou passos.

As experiências relatadas neste estudo mostraram que diferentes analistas podem escrever um grande número de passos complementares ou não de alguma forma, mas a quantidade de passos obrigatórios são menos discrepantes entre os diferentes analistas utilizando a classificação de WAZLAWICK (2004). Porém, com a redução da quantidade de passos nos casos de uso, podem ser realizadas estimativas abaixo do tempo real de desenvolvimento de software. Esta hipótese será tratada no próximo estudo de caso.

---

[4] Média 25%-75% Fora de 25%-75% Fora da Escala

4.2 Estudo de Caso 2 – Avaliação da Subestimação

O Estudo de Caso 2 foi realizado utilizando os casos de uso de um sistema cedido por uma empresa de Blumenau, Santa Catarina, que usa o processo unificado para desenvolver seus softwares. O Sistema possui dois módulos: Financeiro (ANEXO 1) e Secretaria (ANEXO 2); totalizando 13 casos de uso que podem ser visualizados nos anexos.

Foi realizada uma contagem com o método PCU conforme o método padrão de contagem (por etapas ou passos) e depois foi aplicada a classificação dos passos obrigatórios na descrição dos casos de uso sugerido por WAZLAWICK (2004), contabilizando a quantidade de 'Eventos' e 'Respostas de Sistema'.

Os casos de uso foram recebidos já com a classificação dos passos obrigatórios, por isso não foi necessária a identificação dos mesmos, pois este trabalho já tinha sido realizado pela empresa que os cedeu. A empresa fez a contabilidade das horas efetivamente utilizadas para o desenvolvimento, que foi comparada às estimativas.

Outra hipótese deste trabalho seria que, utilizada a classificação dos passos em 'EV' e 'RS' os sistemas poderiam sofrer uma subestimação (*underestimate)*. Com a diminuição dos passos, os casos de uso poderiam não refletir todas as funcionalidades do sistema, fazendo com que o total da estimativa ficasse muito abaixo do tempo real gasto no desenvolvimento. Esperava-se mostrar que a contagem de Eventos e Respostas de Sistema teria uma previsão consistente do esforço. Visto que no primeiro estudo foi mostrado que a contagem de Eventos e Respostas diminuiu no que se refere à contagem feita por diferentes analistas.

O Quadro 10 apresenta os resultados deste estudo de caso. A segunda coluna (Empresa) mostra o número de horas estimadas pela experiência dos analistas da empresa. A terceira coluna (PCU) mostra a estimativa usando o método PCU original, em que todos os passos são contabilizados. A quarta coluna (Obrigatórios) mostra a estimativa quando somente os passos obrigatórios foram contados. A última coluna (Real) são as horas efetivamente empregadas para o desenvolvimento do sistema.

QUADRO 10 - ANÁLISE DOS SISTEMAS

	Empresa	**PCU**	**Obrigatórios**	**Real**
Módulo Financeiro	440,0	647,9	486,3	**450,0**
Porcentagem de erro	-0,02	0,44	0,08	-
Módulo Secretaria	360,0	769,4	456,6	**600,0**
Porcentagem de erro	-0,40	0,28	-0,24	-
Totais	**800**	**1417**	**943**	**1050**
Porcentagem de erro	-0,24	0,35	-0,10	-

A coluna ‘Empresa’ mostra os valores que foram gerados pela empresa, usando o método PCU adaptado pela empresa, usando os fatores de complexidade técnica e os fatores ambientais. Os pesos usados pela empresa não foram informados, somente as estimativas finais foram disponibilizadas. As colunas ‘PCU’ e ‘Obrigatórios’ foram geradas similarmente, a única diferença foram o número de passos considerados pelo caso de uso. Em ambos os casos os seguintes cálculos foram realizados:

a) Os atores foram classificados de acordo com as regras originais do método PCU;

b) Os casos de uso foram classificados conforme o método de contagem utilizado (todos os passos ou somente os passos obrigatórios).

c) Os fatores ambientais foram todos ajustados para Médio (3), visto que não foi passado os valores utilizados na estimativa realizada pela empresa. Os fatores de complexidade técnica foram ignorados como proposto na literatura.

d) A produtividade foi seguida pela proposta de KARNER (1993), com 20 homens/hora por caso de uso.

A linha Porcentagem de Erro ou MRE (Magnitude of Relative Error) apresenta os valores para cada método em relação ao tempo efetivamente gasto para o desenvolvimento (Real).

Este estudo mostrou que o método proposto de descrever os passos dos casos de uso com Eventos e Resposta de Sistemas e classificá-los conforme a quantidade de ‘EV’ e ‘RS’ apresentou melhores resultados que o método PCU original para ambos os

módulos e produziu resultado melhor que a experiência dos analistas da empresa para o segundo módulo e para a soma dos módulos. Porém, as informações para calcular os fatores de complexidade técnica e o valor da produtividade da equipe não foram repassados pela empresa. Com isto, pode-se considerar que talvez os dados utilizados não foram calibrados, ou foram definidos valores que não correspondiam a estimativa da empresa.

Diante destas informações, a hipótese de que a classificação iria gerar estimativas muito abaixo do tempo efetivamente gasto não foi confirmada, mas ao contrário, foram geradas estimativas mais precisas.

4.3 Estudo de Caso 3 – Avaliação do Tempo de Desenvolvimento por Caso de Uso

O Estudo de Caso 3 foi realizado a partir do desenvolvimento de software para uma organização não governamental de Santa Catarina e o objetivo era obter valores preliminares para as variáveis que a teoria proposta necessitava para ser aplicado na fase de concepção. Os valores destas variáveis deverão variar conforme o projeto e a equipe, sendo importante, portanto, caracterizar estes valores.

O estudo de caso consiste em explicar a interferência dos fatores dependentes e independentes no desenvolvimento orientado a objetos, visto que o método de pontos de caso de uso foi desenvolvido como uma evolução do método de Pontos de Função, o qual foi originalmente proposto para o desenvolvimento estruturado.

Neste estudo de caso a equipe era experiente na linguagem de programação e no método de desenvolvimento (tendência para um valor baixo de *TFI*), mas tinha pouca experiência com a ferramenta CASE (tendência para um aumento no *TFI*). O sistema era bem compreendido, a análise era detalhada e um número significativo de casos de uso já estavam listados e expandidos no início do trabalho (tendência de que f seja próxima de 1). O projeto (um sistema de cadastro e emissão de relatórios pela *web*) foi desenvolvido até o final e o produto entregue ao cliente. Em função do tempo real de desenvolvimento deste projeto, os valores das variáveis foram calculados para esta equipe e este projeto.

O tempo total de desenvolvimento TD do sistema foi de 111 horas e cinco minutos. Na fase de concepção, cinco casos de uso foram listados, e ao final do projeto mais um caso de uso foi descoberto. As variáveis da equação (4) foram assim calculadas:

TFI = 23 horas e 49 minutos (tempo gasto com atividades que não dependem da quantidade de casos de uso).

f = 6/5 = 1,2 (fator de aumento do número de casos de uso ao final do desenvolvimento em relação ao valor inicialmente previsto)

$complex(uc) = 1$ para todos os casos de uso.

T = (111h05 – 23h49) / 1,2*5 = 12h07m13 (calculado como $(TD - TFI) / f * N$)

Portanto, nesta etapa calculou-se que o tempo médio para o desenvolvimento dos artefatos derivados de um caso de uso deste tipo de sistema com esta equipe foi de aproximadamente 12 horas e 07 minutos.

Com estes dados apenas, a equação de estimação de esforço da fase de concepção poderia ser definida como:

(9) $TDE = TFI + 12{,}7 * 1{,}2 * N = TFI + 15{,}24 * N$

Para cada caso de uso identificado por esta equipe em futuros sistemas do mesmo tipo, 15,24 horas em média serão necessárias para o desenvolvimento. O valor de TFI não depende do número de casos de uso do sistema e deve, portanto, ser estimado separadamente.

Pressupõe-se que, com o aumento da experiência da equipe esse valor 15,24 vá diminuindo até atingir um mínimo, que corresponde ao máximo desempenho possível da equipe com a tecnologia de desenvolvimento disponível. Investimentos em treinamento, melhoria de processo e automatização aumentarão inicialmente o valor de TFI , mas com o tempo diminuirão o limite mínimo de $T * f$, correspondendo, portanto os ganhos permanentes.

Não foram definidos os valores para a etapa de contratos que é definida pela fórmula 7 e 8 da teoria proposta, porque não foram coletados os dados para este propósito no estudo de caso realizado. Estes dados poderão ser definidos em trabalhos futuros.

O estudo de caso serviu para mostrar que os passos obrigatórios dos casos de uso no desenvolvimento orientado a objetos têm forte influência no valor final do esforço despendido, devido ao encadeamento entre os artefatos produzidos, e que fatores ambientais podem não ser dependentes da complexidade do sistema, como no método PCU original. Os dados deste estudo serviram para definir as variáveis da Teoria para Estimativa de Software Baseada em Casos de Uso no Desenvolvimento Orientado a Objetos definida no capítulo anterior.

5 Conclusões

Com a revisão bibliográfica foram notadas algumas dificuldades em relação à descrição dos casos de uso, por não se ter um método formal para descrevê-los, o que torna esta tarefa um pouco difícil, principalmente para analistas pouco experientes.

Com as pesquisas em relação à descrição dos casos de uso, foi possível conhecer o método sugerido por WAZLAWICK (2004), que classifica os passos dos casos de uso em passos obrigatórios e passos complementares. Os passos obrigatórios incluem os passos que indicam de alguma forma uma troca de informação entre os atores e o sistema. Sem eles o caso de uso não faz sentido, não segue corretamente, ou seja, qualquer analista deverá descrever estes passos. Já os passos complementares, serão escritos em maior ou menor quantidade por diferentes, mas estes passos não têm influência nas estimativas.

A classificação dos passos nos casos de uso sugerida por WAZLAWICK (2004), poderá ajudar analistas menos experientes a descrever os casos de uso, porque independentemente da quantidade de passos complementares que forem adicionados aos casos de uso, somente os passos obrigatórios serão utilizados para gerar as estimativas, uma vez que somente estes passos irão gerar um efetivo esforço para o desenvolvimento do software. Considerando que o primeiro objetivo específico deste trabalho era estudar um método para descrever casos de uso, com este método de classificação sugerido por WAZLAWICK (2004), torna-se possível a descrição dos passos obrigatórios igualmente, pois os analistas tendem a descrever os mesmos passos, tornando-os menos discrepantes, conforme demonstrado no estudo de caso 1.

O segundo objetivo que era melhorar a precisão das estimativas com o método PCU foi evidenciado com o estudo de caso 2, que, utilizando a classificação sugerida por WAZLAWICK (2004), tornou as estimativas mais próximas do real, por não considerar os passos complementares dos casos de uso. Com a redução da contagem dos passos, outra hipótese foi levantada, que seria a subestimação da estimativa, o que não ocorreu no estudo de caso 2.

Com a aplicação do método de desenvolvimento de software orientado a objetos de LARMAN (2002) adaptado por WAZLAWICK (2004) em um projeto piloto, foi possível definir uma teoria que determina a estimativa de esforço em várias fases do processo de análise, diferentemente do método PCU, que apenas aplica a multiplicação dos fatores ambientais sobre a complexidade estimada dos casos de uso e atores. Foi constatado que os diferentes fatores ambientais do método Pontos de Caso de Uso não devem ser tratados apenas como índices (daí a baixa precisão usualmente associada a este método), igualmente ao método de Pontos de Função (VAZQUEZ, 2003), mas como fatores que determinam aumentos ou decréscimos nas diferentes variáveis da equação de estimativa.

A principal contribuição da teoria consiste em gerar estimativas em qualquer fase da análise do desenvolvimento de software, tornando assim esta teoria mais precisa e detalhada ao decorrer de toda a fase da análise. Além disso, a teoria pode ser continuamente ajustada através de dados históricos do desenvolvimento de softwares de cada empresa. Os valores inicialmente definidos poderão ser mais precisos aplicando esta teoria em diferentes projetos de desenvolvimento de software orientado a objeto.

5.1 Limitações do Trabalho

O Estudo de Caso 1 foi realizado com alunos da Pós Graduação em Ciência da Computação da UFSC a fim de comprovar a eficiência do método no que diz respeito à quantidade de passos semelhantes para um mesmo projeto por diferentes analistas.

O Estudo de Caso 2 utiliza dois módulos de um sistema cedido por uma empresa de desenvolvimento de software de Blumenau.

Seria necessária a criação de um histórico de previsões para tornar estes valores mais precisos. Não foi possível aplicar a teoria proposta a um número grande de projetos.

Não foi calculada a precisão das estimativas geradas nos estudos de caso.

5.2 Trabalhos Futuros

Poderá ser estudada uma proposta diferenciada para classificação dos casos de uso, além de Simples, Médio e Complexo, dependendo do número de passos obrigatórios. Também se recomenda um estudo para verificar os pesos relativos aos eventos e respostas de sistema, porque possivelmente os eventos de sistemas terão um esforço maior para serem implementados do que uma resposta de sistema, que geralmente é uma simples consulta.

Os diferentes valores das equações da Teoria para Estimativa de Software Baseada em Casos de Uso no Desenvolvimento Orientado a Objetos podem ser obtidos a partir de uma massa de dados suficientemente grande pela correlação observada e definir valores diferentes para cada tipo de software.

Referências Bibliográficas

ANDA, B.; DREIEM, H.; JORGENSEN M.; SJOBERG, D. (2001) "Estimating Software Development Effort based on Use Cases – Experience from Industry". In M. Gogolla, C. Kobryn (Eds.): UML 2001 - The Unified Modeling Language. Springer-Verlag. 4th International Conference, Toronto, Canada, October 1-5, 2001, LNCS 218.

ANDA, B. (2002) Comparing Effort Estimates Based on Use Case Points with Expert Estimates. Empirical Assessment in Software Engineering (EASE 2002), Keele, UK, April 8-10.

ANDRADE, E.L.P. (2004) Pontos de Caso de Uso e Pontos de Função na gestão de estimativa de tamanho de projetos de software orientados a objetos. Programa de Pós-Graduação em Gestão do Conhecimento e Tecnologia da Informação. Mestrado. Universidade Católica de Brasília. Brasília. 143 p.

ANDRADE, E.L.P. Oliveira, K.M. (2004) Gestão de estimativa de tamanho de projetos de software orientado a objetos. Universidade Católica de Brasília.

BANERJEE, G. (2001) Use Case Points – an estimation approach. Disponível em: <http://www.bfpug.com.br/artigos.htm>. Acesso em: 10/08/2004.

COCKBURN, A. (2005) Escrevendo Casos de Uso Eficazes: um guia prático para desenvolvedores de software. Trad. Roberto Vedoato. Porto Alegre: Bookmam.

FATTOCS. Perguntas e Respostas sobre Análise de Pontos de Função. Disponível em: http://www.fattocs.com.br/faq.asp. Acesso em: 05/2007.

FEITOSA, C.; JANSEN, S. Processo de estimativa e acompanhamento de tamanho e esforço para o ProSCes. In: Encontro da Qualidade e produtividade em software (EQPS), 2003, Fortaleza. Fortaleza, MCT.

FREIRE, Herval. (2003) "Calculando Estimativas: o Método Pontos de Caso de Uso", Developer´s Magazine, nº 78, Fevereiro.

FENTON, N.E.; PFLEEGER, S.L. (1997) Software Metrics. A Rigorous and Practical Approach. Cambridge University Press.

KARNER, Gustav. (1993). Resource Estimation for Objectory Projects. Objectory Systems, September.

KARNER, Gustav. (1993b). Metrics for Objectory. Thesis at the University of Linkopmg, Sweden.LiTH-IDA-Ex-9344;21. December.

KITCHENHAM, B.; KÄNSÄLÄ, K. (1997) Inter-item correllation among function points. National Computing Centre Ltd, UK and VTT, Finland.

LARMAN, Craig. (2002) Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. Prentice-Hall, second edition.

LONGSTREET, D. (2001) Use cases and function points. Copyright Longstreet Consulting Inc. www.softwaremetrics.com.

PRESSMAN, R. S. (2001) Software Engineering: A Practitioner's Approach. 5th Edition, New York: McGraw-Hill.

RIBU, K. Estimating object oriented software projects with use cases. Oslo: University of Oslo, 2001.132 p. Master of Science - Thesis - Department of Informatics.

RULE, P.G. (2001) Using measures to understand requirements. Software Measurement Services Ltd.

SCHNEIDER, G; WINTERS, J. (1998). Applying Use Cases - A Practical Guide. Addison-Wesley.

SMITH, John. (1999) "The estimation of effort based on use cases". Rational Software White Paper.

SPARKS, S.; KASPCZYNSKI, K. (1999) The art of sizing projects. Sun World.

SYMONS, C.R. (1991) Software Sizing and Estimating, MKII FPA. John Wiley and Sons.

VALENÇA, A. (2003) Estimativa de Esforço de Software Orientado a Objetos. Recife. 2003. 43 p.

VAZQUEZ, Carlos Eduardo. (2003) Análise de Pontos de Função: medição, estimativas e gerenciamento de projetos de software. 1. ed. São Paulo: Érica.

VIEIRA, Everton Luiz; WAZLAWICK, Raul Sidnei. (2006) Improving the Use Case Points Method by Counting only Mandatory Steps. Submetido à Revista de Informática Teórica e Aplicada (RITA). Janeiro: JEMS.

VIEIRA, Everton Luiz; WAZLAWICK, Raul Sidnei. (2006) Teoria Explanatória para Estimativa Baseada em Casos de Uso no Desenvolvimento Orientado a Objetos. CLEI - The 32nd Latin-American Conference on Informatics. Chile, Agosto, 2006.

WAZLAWICK, Raul Sidnei. (2004) "Análise e Projeto de Sistemas de Informação Orientado a Objetos". Rio de Janeiro: Elsevier.

WIKIPEDIA. (2006) Source lines of code. Disponível em:

<http://en.wikipedia.org/wiki/Source_lines_of_code>. Acesso em: 10/12/2006.

Referências Consultadas

Antoniol G.; Calzolari F.; Cristoforetti L.; Fiutem, R.; Caldiera, G. (1998) Adapting Function Points To Object Oriented Information Systems. Lecture Notes In Computer Science. Proceedings of the 10th International Conference on Advanced Information Systems Engineering: 59 – 76.

ANDA, B.; MOHAGHEGHI, P; CONRADI, R. (2005) "Effort Estimation of Use Cases for Incremental Large-Scale Software Development". ICSE 2005: St. Louis, MO, USA. Session: Empirical Software Engineering. Pages: 303 - 311.

ANDA, B. (2003) "Empirical Studies of Construction and Application of Use Case Models". Department of Informatics. Faculty of Mathematics and Natural Sciences. University of Oslo, 2003.

ARNOLD M, PEDROSS P. Software Size Measurement and Productivity Rating in a Large-Scale Software Development Department. Disponível em: <http://www.bfpug.com.br/artigos.htm>. Acesso em: 05/2004.

BANERJEE, G. Use Case Estimation - Framework. Annual IPML Conference, 2004. Disponível em: <www.qaiindia.com/Conferences/presentations/gautam_birlasoft_pre.pdf>. Acesso em: 10/08/2004.

BARONI, A.L. (2002) Formal Definition of Object-Oriented Design Metrics. Faculty of Sciences In Collaboration with Ecole des Mines de Nantes – France and Universidade Nova de Lisboa – Portugal. Disponível em: <http://ctp.di.fct.unl.pt/QUASAR/Resources/Papers/2002/ThesisAline.pdf>. Acesso em: 03/2007.

CALDIERA G, LOKAN C, ANTONIOL G, FIUTEM R, CURTIS S, COMMARE GL e MAMBELLA E. Estimating Size and Effort for Object Oriented System. Disponível em: <http://citeseer.ist.psu.edu/456963.html>. Acesso em: 02/2005.

COCKBURN, A. Use Case Fundamentals. Disponível em: <http://alistair.cockburn.us/index.php/Use_case_fundamentals>. Acesso em: 02/2006.

COCKBURN, A. Use Cases in Theory & Practice. Disponível em: <www.cc.gatech.edu/classes/AY2001/cs3300_spring/docs/usecasetalk.ppt>. Acesso em: 06/2005a.

COCKBURN, A. Structuring use cases with goals. Disponível em: <http://alistair.cockburn.us/index.php/Structuring_use_cases_with_goals>. Acesso em: 07/2005b.

COCKBURN, A. Writing Effective Use Cases. Disponível em: <http://alistair.cockburn.us/crystal/books/weuc/weuc0002extract.pdf.>. Acesso em: 04/2005c.

COX, K. Cognitive Dimensions of Use Cases - feedback from a student questionnaire. Empirical Software Engineering Research Group School of Design, Engineering and Computing Bournemouth University. Disponível em: <www.ppig.org/papers/12th-cox.pdf> Acesso em: 05/2006.

DAMODARAN, M. Estimation Using Use Case Points. Disponível em: <http://www.bfpug.com.br/artigos.htm>. Acesso em: 05/2004.

JACOBSON, I. Use Cases - Yesterday, Today, and Tomorrow. Disponível em: <http://www.therationaledge.com/content/mar_03/f_useCases_ij.jsp>. Acesso em 07/2004.

JØRGENSEN M., KIRKEBØEN G., SJØBERG D. I., ANDA B., BRATTHALL L. (2000) Human Judgement in Effort Estimation of Software Projects. Disponível em: <http://www.simula.no/research/engineering/publications/SE.5.Joergensen.2000.b>. Acesso em: 05/2004.

NAGESWARAN, S. Test Effort Estimation Using Use Case Points. Quality Week 2001, San Francisco, California, USA, June 2001. Disponível em: <http://www.bfpug.com.br/artigos.htm>. Acesso em: 05/2004.

ROBERTS, F.S. History of Software Measurement. Disponível em:

<http://irb.cs.tu-berlin.de/~zuse/metrics/3-hist.html>. Acesso em: 02/2005.

Apêndice

Exemplo didático com alguns passos do método de desenvolvimento de software adaptado por Wazlawick (2004) e estimativa com o método Pontos de Caso de Uso. Conforme segue a relação:

- **Sumário Executivo**
- **Levantamento de Requisitos - Funcionais e Não-Funcionais**
- **Organização dos requisitos em Casos de Uso**
- **Organização dos requisitos em Função de Conceitos**
- **Organização dos requisitos em Consultas**
- **Expansão dos Casos de Uso**
- **Estimativa com o método Pontos de Caso de Uso**

Abaixo um exemplo do Sistema de Biblioteca adaptado de Wazlawick(2004):

Sumário Executivo

O sistema irá controlar o empréstimo e a devolução de livros da biblioteca, bem como as reservas e cancelamento de reservas. O sistema deverá fazer a cobrança de multas pelo atraso na devolução dos livros.

Levantamento de Requisitos - Funcionais e Não-Funcionais

F1. Cadastrar Livro		**( ) Oculto**		
Descrição: o sistema deve permitir o cadastro dos livros, indicando o título, autor, edição, editora, cidade e ano de publicação, palavras-chave, número de páginas.				
Requisitos Não-Funcionais				
Nome	Descrição	Categoria	Desejável	Permanente
NF 1.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de bibliotecário(a).	Segurança		X
NF 1.2 Descrição das palavras-chave	No campo das palavras-chave as mesmas devem ser separadas por ponto-e-vírgula.	Interface		X

F2. Cadastrar Funcionário		**( ) Oculto**		
Descrição: o sistema deve permitir o cadastro dos funcionários, indicando o nome, endereço completo, RG, CPF, data de nascimento, telefones, e-mail, perfil, data de entrada e data de saída.				
Requisitos Não-Funcionais				
Nome	Descrição	Categoria	Desejável	Permanente
NF 2.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de coordenador.	Segurança		X
NF 2.2 Validação do CPF	A função deve validar o CPF conforme informações do governo	Segurança		X

F3. Cadastrar Cliente		**( ) Oculto**		
Descrição: o sistema deve permitir o cadastro dos clientes, indicando o nome, endereço completo, RG, CPF, data de nascimento, telefones, e-mail, cargo e situação (normal, pendente, inativo).				
Requisitos Não-Funcionais				
Nome	Descrição	Categoria	Desejável	Permanente
NF 3.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de coordenador, bibliotecário(a) ou atendente.	Segurança		X
NF 3.2 Validação do CPF	A função deve validar o CPF conforme informações do governo	Segurança		X

F4. Registrar Empréstimo	**( ) Oculto**

Descrição: o sistema deve registrar o empréstimo dos livros, indicando o cliente e o(s) livro(s) que foram emprestados, a data de empréstimo, o funcionário que fez empréstimo e a data de devolução.

Requisitos Não-Funcionais

Nome	Descrição	Categoria	Desejável	Permanente
NF 4.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de bibliotecário(a) ou atendente.	Segurança		X
NF 4.2 Identificação do cliente	A identificação do cliente deve ser feita através de uma carteirinha com código de barras.	Interface		X
NF 4.3 Identificação do Livro	A ident. do livro deve ser feita através de um código de barras localizado na contra capa do livro.	Interface		X
NF 4.4 Identificação do Funcionário	A identificação do func. deve ser feita através do código e nome.	Interface		X

F5. Registrar Devolução	**( ) Oculto**

Descrição: o sistema de registrar a devolução(ões), indicando o cliente e os livros devolvidos, bem como comparar as datas de devolução com a data atual, gerar débito(s) para o cliente e marcar cliente como pendente.

Requisitos Não-Funcionais

Nome	Descrição	Categoria	Desejável	Permanente
NF 5.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de bibliotecário(a) ou atendente.	Segurança		X
NF 5.2 Identificação do cliente	A identificação do cliente deve ser feita através de uma carteirinha com código de barras.	Interface		X
NF 5.3 Identificação do Livro	A identificação do livro deve ser feita através de um código de barras localizado na contra capa do livro.	Interface		X
NF 5.4 Multa	A cada livro devolvido deve ser comparadas a data de devolução prevista e a data atual, caso houver atraso, gerar multa para o cliente e marcar o cliente como pendente.	Interface		X

F6. Registrar Pagamento	**( ) Oculto**

Descrição: o sistema deve mostrar o(s) débito(s) do cliente e dar baixa da conta do cliente.

Requisitos Não-Funcionais

Nome	Descrição	Categoria	Desejável	Permanente
NF 6.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de bibliotecário(a) ou atendente.	Segurança		X
NF 6.2 Identificação do cliente	A identificação do cliente deve ser feita através de uma carteirinha com código de barras.	Interface		X

F7. Reservar Livro	**( ) Oculto**

Descrição: o sistema deve permitir a reserva do livro que estiver emprestado, indicando o cliente e o livro.

Requisitos Não-Funcionais

Nome	Descrição	Categoria	Desejável	Permanente
NF 7.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de bibliotecário(a) ou atendente.	Segurança		X
NF 7.2 Identificação do cliente	A identificação do cliente deve ser feita através de uma carteirinha com código de barras.	Interface		X
NF 7.3 Identificação do Livro	A identificação do livro deve ser feita através de um código de barras localizado na contra capa do livro.	Interface		X
NF 7.4 Enviar e-mail	Quando for feita a devolução do livro, enviar um e-mail para o cliente que reservou o livro.	Especificação		

F8. Gerar Relatórios	**( ) Oculto**

Descrição: o sistema deve permitir a impressão dos relatórios dos livros, funcionários, clientes, reservas e clientes pendentes.

Requisitos Não-Funcionais

Nome	Descrição	Categoria	Desejável	Permanente
NF 8.1 Controle de acesso	A função deve permitir somente o acesso a usuário com perfil de bibliotecário(a) ou atendente.	Segurança		X

Organização dos requisitos em Casos de Uso

Nome	**Atores**	**Descrição**	**Referências cruzadas**
Emprestar Livros	Cliente, Funcionário	O cliente se identifica e identifica os livros que deseja levar. O funcionário faz o registro e libera os livros para empréstimo.	F1, F2, F3, F4, F6
Devolver Livros	Cliente, Funcionário	O cliente entrega ao funcionário os livros. O funcionário faz o registro da devolução e o cliente efetua o pagamento devido.	F1, F2, F3, F4, F5
Reservar Livros	Cliente, Funcionário	O cliente solicita a reserva de um ou mais livros. O funcionário registra a reserva.	F1, F2, F3, F4, F7

Organização dos requisitos em Função de Conceitos

Conceito	I	A	E	C	Observações	Ref. Cruzadas
Livro	x	x	x	x	Somente livros sem empréstimos podem ser excluídos	F1
Funcionário	x	x	x	x		F2
Cliente	x	x	x	x	Somente clientes sem empréstimos podem ser excluídos	F3
Empréstimo			x	x	Pode ser incluído apenas através do caso de uso “Emprestar Livros”. Não pode ser alterado, apenas excluído.	F4, F5
Reserva			x	x	Pode ser incluído apenas através do caso de uso “Reservar Livros”. Não pode ser alterado, apenas excluído.	F7

As letras acima referem-se a Inserção (I), Alteração (A), Exclusão (E) e Consulta (C).

Organização dos requisitos em Consultas

Conceito	Ref. Cruzadas
Livros Cadastrados	F1
Funcionários Registrados	F2
Clientes Cadastrados	F3
Empréstimos do Mês	F1, F3, F4
Livros com Devolução Atrasada – Clientes Pendentes	F1, F3, F4, F5
Reservas Cadastradas	F7

Casos de Uso Expandidos

Caso de Uso: Emprestar Livros
Atores: Cliente, Funcionário.
Pré-condições: Os livros a serem emprestados devem estar devidamente cadastrados e identificados.
Pós-condições: O registro do empréstimo dos livros deve ser finalizado e o cliente deve ficar de posse dos mesmos. O cliente deve ser informado a respeito do prazo e valores relativos à devolução.
Requisitos Correlacionados: F4 e F6
Fluxo principal: 1. O cliente seleciona os livros a serem emprestados; 2. O cliente informa ao funcionário os livros a serem emprestados; 3. [EV] O funcionário identifica o cliente e inicia o empréstimo; 4. [EV] O funcionário registra cada um dos livros para empréstimo; 5. [RS] O funcionário finaliza o empréstimo e informa ao cliente a respeito dos prazos e valores relativos a devolução;
Fluxos Alternativos: **3.a. O cliente não possui cadastro;** 3.a.1. O cliente informa seus dados; 3.a.2. [EV] O funcionário realiza seu cadastro; 3.a.3. Retorna ao passo 3; **3.b. O cliente possui débitos vinculados ao seu cadastro;** 3.b.1. O cliente paga o seu débito; 3.b.2. [EV] O funcionário registra a quitação e elimina o débito existente; 3.b.3. Retorna ao passo 3; **4.a. O livro está reservado para outro cliente;** 4.a.1. [RS] O funcionário informa que o livro não está disponível para empréstimo; 4.a.2. Retorna ao passo 4; **4.b. O livro está danificado;** 4.b.1. [EV] O funcionário registra que o livro está danificado; 4.b.2. O funcionário verifica se existe algum outro exemplar do mesmo livro disponível; 4.b.3. Retorna ao passo 4;

Caso de Uso: Devolver Livros
Atores: Cliente, Funcionário.
Pré-condições: Os livros a serem devolvidos devem estar devidamente cadastrados, identificados e emprestados.
Pós-condições: O cliente deve devolver os livros emprestados e o registro de devolução deve ser finalizado. O cliente deve ser informado a respeito de débitos, caso existam.
Requisitos Correlacionados: F5 e F6.
Fluxo principal: 1. O funcionário recebe os livros a serem devolvidas; 2. [EV] O funcionário registra a devolução de cada livro; 3. [RS] O funcionário finaliza a devolução e informa ao cliente a respeito de multas e ressarcimentos sobre a(s) devolução(ões); 4. O cliente vai embora com o comprovante de devolução;
Fluxos Alternativos: **1.a. O livro entregue está danificado;** 1.a.1. [EV] O funcionário registra que o livro está danificado; 1.a.2. O funcionário registra o débito para o cliente; 1.a.4. Retorna ao passo 2; **2.a. O cliente possui débitos vinculados ao seu cadastro;** 2.a.1. O funcionário informa o cliente a respeito do atraso e sobre a existência de débitos; 2.a.2. [EV] O funcionário registra o pagamento das pendências e as elimina; 2.a.3. Retorna ao passo 4;

Caso de Uso: Reservar Livros
Atores: Cliente, Funcionário.
Pré-condições: Os livros a serem reservados e o cliente devem estar devidamente cadastrados e identificados.
Pós-condições: O registro do reserva dos livros deve ser finalizado.
Requisitos Correlacionados: F7
Fluxo principal: 1. O cliente informa ao funcionário os livros a serem reservados; 2. [EV] O funcionário identifica o cliente e inicia o reserva; 3. [EV] O funcionário registra cada um dos livros a ser reservado; 4. [RS] O funcionário finaliza a reserva e informa ao cliente o prazo de validade;
Fluxos Alternativos: **3.a. O livro está emprestado;** 3.a.1. O funcionário informa ao cliente que o livro está emprestado; 3.a.2. Retorna ao passo 3;

Estimativa com o método Pontos de Caso de Uso

Seguindo o processo de contagem do método pontos de caso de uso, apresentado no capítulo 2.2.4, o primeiro passo é a classificação dos atores, que seguem abaixo:

- UAW (Unadjusted Actor Weight) é calculado pela contagem do total de atores em cada categoria, multiplicando cada total pelo seu específico fator de peso e adicionando os produtos.

Classificação de Ator			
Ator	**Peso**	**Quantidade**	**Total UAW**
Complexo	3	2	6
UAW			**6**

- Os casos de uso são classificados dependendo do número de transações envolvidas no processo. O total de transações de um caso de uso é a soma das transações do fluxo principal com os fluxos alternativos. Os UUCW (Unadjusted Use Case Weights) são calculados pela contagem do número de casos de uso em cada categoria, multiplicando cada categoria de caso de uso com seus pesos e adicionando os produtos. O UAW é adicionado ao UUCW para determinar o UUPC (Unadjusted Use Case Points).

Classificação do Caso de Uso			
Caso de Uso	**Peso**	**Quantidade**	**Total**
Simples	5	1	5
Médio	10	2	20
UUCW			**25**

UUPC = UAW + UUCW
UUPC = 6 + 25 = 31

- Os pontos de caso de uso são ajustados com base nos valores determinados conforme os fatores de complexidade técnica e os fatores ambientais. Em cada

fator é determinado um valor entre 0 e 5, dependendo da influência no projeto. Nos dois casos, fatores técnicos e fatores ambientais, se todas as influências forem iguais a 3, então estes fatores não influenciarão no resultado da estimativa, pois nos dois casos as fórmulas terão aproximadamente o valor igual a 1. Este será o valor utilizado por ser um exemplo didático.

- Os PCU (pontos de caso de uso) são definidos conforme o resultado da fórmula UUCP * TCF * EF. Como nossos valores de ajustes não terão influência, então temos o mesmo valor que UUPC que é 31.

- Para calcular a produtividade com o método pontos de caso de uso, utilizaremos 20 horas/homem por PCU, conforme definido experimentalmente por KARNER(1993).

Concluindo, o Sistema de Biblioteca aqui especificado, levaria 620 horas para ser desenvolvido. Que poderíamos calcular em dias, ou seja, 620 horas dividido por 8 horas diárias, chegando ao resultado de aproximadamente 78 dias ou pouco mais de 2 meses e meio.

Anexo 1

Nº do caso de uso	CU 01 – Gerar Arrecadação de Anuidade
Atores primários	Tesoureiro
Atores secundários	Banco, Inscrito, Sistema.
Fluxo principal	1. O tesoureiro acessa a geração de cobrança de anuidade do sistema. 2. O sistema exibe a tabela de valores de anuidade por faixa de pessoas. 3. O tesoureiro confere a tabela e verifica os valores que serão gerados para os inscritos. 4. [EV] O tesoureiro solicita ao sistema a geração do arquivo de cobrança. 5. [RS] O sistema gera o arquivo no formato solicitado. 6. O tesoureiro envia o arquivo ao banco para que este imprima os boletos e envie aos inscritos.
Fluxos alternativos e exceções	3.1 O tesoureiro poderá alterar a tabela de anuidades neste passo.

Nº do caso de uso	CU 02 – Gerar Arrecadações Gerais
Atores primários	Tesoureiro
Atores secundários	Sistema, Cliente do Conselho.
Fluxo principal	7. O tesoureiro acessa a geração de cobrança de taxas do sistema. 8. [EV] O tesoureiro preenche os dados para a geração da taxa. 9. [RS] Sistema oferece opção para impressão do boleto bancário.
Fluxos alternativos e exceções	3.1 Caso o tesoureiro desejar parcelar uma taxa, ele deve repetir o fluxo principal n vezes, onde n corresponde o número de parcelas.

Nº do caso de uso	CU 03 – Receber Arrecadação.
Atores primários	Tesoureiro
Atores secundários	Sacado, Sistema.
Fluxo principal	10. [EV] Tesoureiro localiza o sacado pelo processo padrão. 11. [RS] Sistema apresenta as cobranças pendentes do sacado. 12. [EV] Sacado informa qual das cobranças deseja pagar. 13. [EV] Tesoureiro recebe o dinheiro ou cheque e informa o valor recebido ao sistema. 14. [RS] Sistema imprime recibo do pagamento. 15. Tesoureiro entrega o recibo ao sacado.
Fluxos alternativos e exceções	4.1 [EV] Se o pagamento for com cheque, sistema irá solicitar o cadastro do cheque.

Nº do caso de uso	CU 04 – Registrar Recebimento Automático.
Atores primários	Tesoureiro
Atores secundários	Banco, Sistema.
Fluxo principal	16. [EV] Tesoureiro envia o arquivo para o sistema. 17. [RS] Sistema lê o arquivo e faz as baixas das cobranças.
Fluxos alternativos e exceções	

Nº do caso de uso	CU 05 – Registrar Pagamento de contas.
Atores primários	Tesoureiro
Atores secundários	Sistema.
Fluxo principal	18. [EV] Tesoureiro acessa o sistema. 19. [RS] Sistema avisa que existem contas vencendo neste dia. 20. [EV] Tesoureiro seleciona as contas que deseja pagar. 21. [RS] Sistema imprime os cheques. 22. [RS] Sistema imprime a cópia dos cheques.
Fluxos alternativos e exceções	

Nº do caso de uso	CU 06 – Fazer conciliação bancária automática.
Atores primários	Tesoureiro
Atores secundários	Sistema.
Fluxo principal	23. Tesoureiro recebe arquivo (com layout definido) do banco. 24. [EV] Tesoureiro envia arquivo ao sistema. 25. [RS] Sistema mostra o resumo das alterações.
Fluxos alternativos e exceções	

Nº do caso de uso	CU 07 – Fazer conciliação bancária manual.
Atores primários	Tesoureiro
Atores secundários	Sistema.
Fluxo principal	26. Tesoureiro verifica o extrato bancário. 27. [EV] Tesoureiro localiza o cheque emitido no sistema. 28. [EV] Tesoureiro informa a data que este entrou na conta. 29. [RS] Sistema atualiza sua base de dados.
Fluxos alternativos e exceções	

Anexo 2

Nº do caso de uso	CU 01 – Manter Requerimentos
Atores primários	Secretário
Atores secundários	Requerente
Fluxo principal	O secretário informa ao sistema tipo de requerimento (processo) e o tipo de pessoa (PF, PJ) solicitado pelo requerente. [EV] O sistema apresenta os documentos necessários para a realização do requerimento; O secretário informa ao sistema quais documentos o requerente está apresentando no momento. O secretário preenche os dados do requerimento necessários. [EV] O sistema grava o requerimento e gera um comprovante para impressão. Secretário imprime o comprovante e entrega ao requerente.
Fluxos alternativos e exceções	No passo 1, o secretario poderá consultar, alterar ou excluir um requerimento localizando-o pelo número do conselho, número do requerimento, nome do requerente ou tipo de requerimento (PF, PJ); No passo 2, caso o requerente já for um corretor cadastrado o secretário poderá consultar os dados do corretor, ao invés de preencher todos os campos novamente No passo 3, caso o tipo de requerimento exigir a cobrança de taxa, será lançado um débito para o requerente.

Nº do caso de uso	CU 02 – Tramitar requerimentos
Atores primários	Usuário (Qualquer papel do conselho)
Atores secundários	Sistema
Fluxo principal	1.Usuário localiza o requerimento através do número do requerimento, número do conselho regional, nome do requerente e/ou tipo do requerimento (PF, PJ); 2.[EV] Sistema lista os requerimentos do requerente que estão em tramite; 3.Usuário seleciona o requerimento; 4.Sistema exibe o requerimento listando também o(s) passo(s) pelo qual o requerimento já tramitou e qual (is) o(s) próximo (s) passo(s) que ele deve tramitar; 5.Usuário confirma ou altera o próximo passo informando o parecer do tramite. 6.[EV] Sistema grava a alteração.
Fluxos alternativos e exceções	1. No passo 5, caso o próximo tramite for o de fechamento do requerimento o sistema deverá fechar o requerimento de acordo com o seu tipo (assunto).

Nº do caso de uso	CU 03 – Enviar correspondências
Atores primários	Secretário
Atores secundários	Sistema
Fluxo principal	7.Secretário informa os dados da correspondência a ser enviada no sistema e seleciona o seu tipo; 8.[EV] O sistema cria um número seqüencial de registro de correspondência conforme as regras estabelecidas (ver documento de requisitos); 9.Secretário cria um documento, no Word ou outro programa equivalente, referente à correspondência, colocando no documento o número de registro da correspondência gerada pelo sistema; 10.Secretário envia o documento ao sistema; 11.[EV] Sistema anexa o documento ao cadastro da correspondência e informa mensagem de confirmação.
Fluxos alternativos e exceções	Não há.

Nº do caso de uso	CU 04 – Receber correspondências
Atores primários	Secretário
Atores secundários	Sistema, Superintendente
Pré-condições	Secretário deve estar identificado pelo sistema. Secretário dever ter recebido as correspondências enviadas ao conselho.
Fluxo principal	12.O secretário abre a correspondência; 13.[EV] O secretário cadastra a correspondência no sistema informando o nome do remetente, data de entrada, local de origem e tipo da correspondência; 14.[RS] O sistema gera um documento para impressão com os dados da correspondência e o número de registro da correspondência; 15.[EV] O secretário encaminha a correspondência fazendo o primeiro tramite; 16.O Secretário imprime o documento; 17.O Secretário anexa o documento impresso a correspondência física e encaminha à superintendência para conhecimento e despacho (passo este não automatizado pelo sistema).

Nº do caso de uso	CU 05 – Tramitar correspondências
Atores primários	Usuário (Qualquer papel do conselho)
Atores secundários	Sistema
Pré-condições	1. Usuário deve estar identificado pelo sistema; 2. Usuário deve possuir a correspondência física em mãos; 3. A correspondência deve estar registrada no sistema;
Fluxo principal	18.[EV] Usuário consulta a correspondência no sistema pelo numero de registro e/ou pelo departamento; 19.[RS] Sistema apresenta os dados da correspondência sem opção para alteração; 20.[EV] Usuário informa o próximo destino da correspondência (sessão) e o motivo do tramite ou parecer; 21.Usuário envia a correspondência física para o seu destino (sessão);

Nº do caso de uso	CU 06 – Manter corretores
Atores primários	Secretário
Atores secundários	Sistema, Corretor
Pré-condições	1. Secretário deve estar identificado pelo sistema; 2. Para a inclusão de um novo corretor o mesmo deve ter passado pelo processo de requerimento de inscrição CU 01;
Fluxo principal	1.Secretário seleciona a opção de incluir um novo corretor no sistema; 2.[EV] Secretário Informa o número do requerimento de inscrição (protocolo); 3.[RS] Sistema importa os dados do requerimento de inscrição e apresenta os dados na tela para o Secretário; 4.[EV] O Secretário faz as alterações e complementações necessárias no cadastro de pessoa; 5.[EV] O Secretario informa os dados de endereço (comercial, residencial) e de contato (fone, fax e-mail etc) do corretor; 6.[EV] O Secretario informa os dados adicionais; 7.[EV] Sistema, grava o corretor, apresentando mensagem de confirmação e gera o número do CRECI;
Fluxos alternativos e exceções	1.**Alterar**. a.Corretor solicita a alteração cadastral para o Secretário, ou através de requerimento; b.[EV] O Secretário consulta os dados do corretor no sistema; c.[EV] O Secretário altera as informações solicitadas pelo corretor e confirma; d.[EV] Sistema mostra mensagem de confirmação; 2.**Imprimir Certificados (7)** de Inscrição para pessoas jurídicas e carteira de identificação profissional para pessoas físicas após o cadastro dos mesmos através de um link, ou através de atalho após uma busca; 3.**Opção consulta somente leitura** após a busca de um corretor deve-se apresentar a opção de consulta somente leitura para que o secretário apenas consulte os dados de uma pessoa sem correr o risco de